Escola Politècnica Superior
de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FIN DE CARRERA

TÍTULO: Control de un módulo bluetooth mediante microcontrolador

AUTOR: Jaime Gálvez Navarro

DIRECTOR: Gabriel Montoro López

SUBDIRECTOR: José Luís Valenzuela González

FECHA: 25 de febrero de 2005

Título: Control de un módulo bluetooth mediante microcontrolador

Autor: Jaime Gálvez Navarro

Titulación: Ingeniería Técnica de Telecomunicación

Especialidad: Sistemas de Telecomunicación

Plan: 2000

Director: Gabriel Montoro López

Departamento: Teoría de la señal y comunicaciones

Subdirector: José Luís Valenzuela González

Departamento: Teoría del la señal i comunicaciones

Conformidad,

Director del TFC

TÍTULO: Control de un módulo bluetooth mediante microcontrolador

AUTOR: Jaime Gálvez Navarro

DIRECTOR: Gabriel Montoro López

SUBDIRECTOR: José Luís Valenzuela González

FECHA: 25 de febrero de 2005

Resumen

El estándar bluetooth es una norma abierta que posibilita la conexión inalámbrica de corto alcance de voz y datos entre ordenadores, portátiles, agendas digitales personales, teléfonos móviles, impresoras, escáneres, cámaras digitales e incluso dispositivos de casa, a través de una banda disponible a nivel global (2,4 GHz) y mundialmente compatible. Es decir, la tecnología bluetooth es el sistema de comunicaciones sin hilos del futuro, el cual elimina el engorroso lío de cables de comunicación entre los diferentes dispositivos electrónicos.

Con la idea de conseguir una comunicación de este estilo, el objetivo de este trabajo final de carrera ha sido poder controlar un módulo bluetooth mediante un microcontrolador de manera que podamos comunicar un ordenador y un microcontrolador utilizando este tipo de tecnología.

Los materiales utilizados para realizar este proyecto han sido 2 módulos ROK 101 008 de Ericsson (bluetooth), un ordenador convencional y un microcontrolador del tipo pic18f452 de la casa Microchip.

Forma de montaje:
Uno de los módulos de bluetooth se conecta al PC mediante el puerto de comunicaciones COM (utilizando el estándar RS232) y el otro módulo se conecta al microcontrolador (también mediante el estándar RS232) y de esa manera se procede a establecer comunicación entre el PC y el microcontrolador vía bluetooth.

Para realizar este montaje ha sido necesario familiarizarse con las características y la forma de programación de la familia de microcontroladores de Microchip así como aprender a utilizar la especificación del estándar bluetooth y RS-232.

TITLE: Control of a module bluetooth with a microcontroller

AUTHOR: Jaime Gálvez Navarro

DIRECTOR: Gabriel Montoro López

ASSISTANT DIRECTOR: José Luís Valenzuela González

DATE: February, 25th 2005

Abstract

The standard bluetooth is an open norm that makes possible the wireless connection of short reach of voice and data between computers, laptops, personal digital agendas, mobiles, printers, scanners, digital cameras and even home devices, through a band available at global level (2.4 GHz) and world-wide compatible. That is to say, the technology bluetooth is the system of communications without threads of the future which eliminates the troublesome cable mess of communication between the different electronic devices.

With the idea to obtain a communication of this style, the final objective of this TFC has been to be able to control a module bluetooth by means of a microcontroller so that we could communicate a computer and a microcontroller using this type of technology.

The used materials to make this project have been 2 modules ROK 101 008 of Ericsson (bluetooth), a conventional computer and a microcontroller type pic18f452 of the Microchip enterprise.

Form of assembly:
One of the modules of bluetooth connects to the PC through the COM communications port (using standard RS232) and the other module connects the microcontroller (also by standard RS232) and from that way it is established contact between the PC and the microcontroller by bluetooth.

In order to make this assembly it has been necessary to learn the characteristics and how to program this family of Microchip microcontrollers as well as to learn to use the specification of the standard bluetooth and RS-232.

Este TFC simboliza el final de mi vida universitaria, por ello quiero dedicárselo a mi familia y amigos, que sin su ayuda y comprensión, no habría podido finalizar mis estudios. También quiero hacer un pequeño recordatorio a todos los profesores que me han impartido clases por su gran esfuerzo y dedicación, ya que sin su aportación no habría adquirido ni los conocimientos ni las ganas de de superarme que hoy tengo.

Gracias a todos.

ÍNDICE.

INTRODUCCIÓN.

Actualmente existen en el mercado una serie de tecnologías mediante las cuales podemos enviar información o incluso realizar conexiones entre diferentes aparatos electrónicos sin la necesidad del uso de cables, los cuales funcionan de forma eficiente pero su instalación y configuración es bastante engorrosa.

Hoy en día la sociedad tiende a informatizarse cada vez más y ello conlleva la necesidad de tener que interconectar los diferentes aparatos que existen en el mercado de una forma fácil y sencilla.

En el mercado ya existen tecnologías inalámbricas como IrDa, pero aspectos como su ancho de banda o su operatividad (tiene que haber línea visual entre los dispositivos que se quieren comunicar) no han permitido el asentamiento esperado en el mercado. Por ello, se llegó a la conclusión que la solución era un sistema vía radio de bajo coste que permitiera la conectividad entre dispositivos en ausencia de cables y sin la necesidad de que hubiera línea visual entre emisor y receptor.

Actualmente el Bluetooth es una tecnología en auge que ofrece muchas ventajas respecto al resto de tecnologías y en la que parece que todas las empresas del sector de las telecomunicaciones y la informática están muy interesadas.

Objetivos y organización del trabajo:

Con el objetivo de hacer un estudio de la tecnología Bluetooth en este TFC se pretende realizar una conexión simple entre un PC convencional y un microcontrolador mediante tecnología Bluetooth. Para ello, hemos creído necesario dividir este trabajo en 6 partes fundamentales para el buen entendimiento del mismo:

- En el capítulo 1 se da una visión global de lo que es la tecnología bluetooth y cuales son sus principales características y funcionamiento.

- En el capítulo 2 se da una visión global de lo que son los microcontroladores PIC y cuales son sus principales características, en especial las del microcontrolador PIC 18f452 de Microchip.

- En el capítulo 3 se ponen en práctica todos los conceptos aprendidos y se realizara una conexión entre un PC y un microcontrolador de manera que podamos establecer una conexión Bluetooth, y si es posible, enviar información entre ellos.

Finalmente en los capítulos 4, 5 y 6 expondremos las conclusiones finales así como los criterios medioambientales y las líneas futuras más relevantes.

Para concluir este TFC, en la parte final se puede encontrar un anexo con una guía rápida del MPLAB que queremos resaltar para programar el PIC.

Capítulo 1. Bluetooth.

En este apartado, queremos dar una visión global de lo que es el Bluetooth, haciendo una pequeña introducción a nivel básico de cual es su funcionamiento y así posteriormente poder comprender los conceptos del TFC que nos ocupa.

1.1. ¿Qué es el Bluetooth?

El Bluetooth es una tecnología orientada a la conectividad inalámbrica entre dispositivos tan dispares como PCs, PDAs, teléfonos móviles, electrodomésticos, etc. En general, podemos decir que las posibilidades pueden considerarse infinitas.

El Bluetooth, a parte de ser una nueva tecnología, es también una especificación abierta para comunicaciones inalámbricas de voz y datos. Está basado en un enlace de radio de bajo coste y corto alcance, el cual proporciona conexiones instantáneas (adhoc) tanto para entornos de comunicaciones móviles como estáticos.

Esta tecnología ha revolucionado el mercado de la conectividad ya que es capaz de comunicar cualquier dispositivo que cumpla con las especificaciones inalámbricas del Bluetooth.

La principal ventaja que ofrece esta tecnología es la conectividad sin cables de todos los dispositivos, pero más que reemplazar los incómodos cables, esta tecnología ofrece un puente entre las redes de datos hoy existentes y el exterior.

El Bluetooth, al ser un estándar abierto, pretende conectar una amplia gama de dispositivo sin importar su marca. Sus principales características son:

- Robustez.
- Bajo coste.
- Necesidad de poca potencia.
- Baja complejidad.
- Es un estándar global.

La tecnología Bluetooth comprende hardware, software y requerimientos de interoperatibilidad, por lo que para su desarrollo ha sido necesaria la participación de los principales fabricantes de los sectores de las telecomunicaciones y la informática.

En la siguiente figura (fig.1) podemos ver una imagen global de lo que pretende esta especificación.

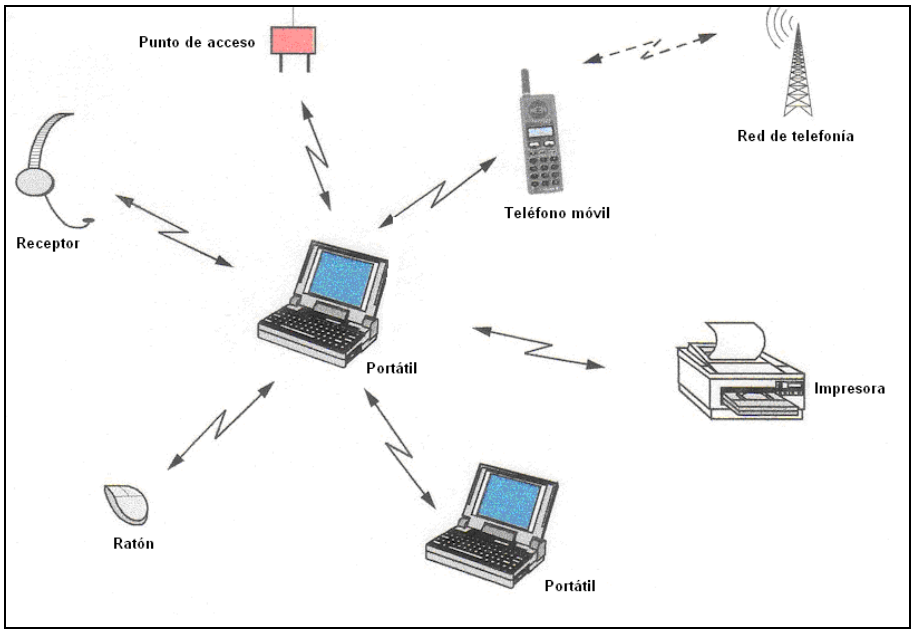


Fig.1.- Comunicación bluetooth

1.2. Historia del Bluetooth.

El nombre Bluetooth es en honor a un rey Danés del siglo X llamado Harald Blaatand, fue rey de Dinamarca entre los años 940-981 D.C. De este rey se dice que tenía unas grandes habilidades comunicativas que lo hicieron famoso y con las cuales inició el proceso de cristianización de la sociedad vikinga.

La historia del bluetooth es relativamente corta. Las primeras investigaciones fueron realizadas por Ericsson Mobile Communications en 1994. Esta compañía promovió una iniciativa para estudiar la viabilidad de una interfaz por radio entre los teléfonos móviles y sus accesorios, con la característica y condición de que tuviese un bajo coste y pequeño consumo. Hasta entonces, las tecnologías de comunicaciones basadas en el cable funcionaban con eficiencia, sin embargo, su instalación y configuración resultaba bastante dificultosa.

Conforme avanzaba el proyecto de Ericsson fue despertando el interés de otros fabricantes y enseguida se vio el gran abanico de posibilidades que ofrecía esta tecnología, así que a principios de 1998 se creo la SIG (grupo de interés especial) el cual estaba formado por Ericsson, Nokia, IBM, Toshiba e Intel. Actualmente se han ido añadiendo otras empresas al grupo como 3Com Corporation, Lucent Technologies, Microsoft Corporation, Motorota Inc y otras muchas más llegando hoy en día a estar formado por más de 2000 empresas del sector de la informática y las telecomunicaciones.

El propósito principal del SIG es establecer un standard para la interface aérea junto con su software de control, su fin es asegurar la interoperatibilidad de los diversos equipos de diferentes fabricantes.

1.3. ¿Cómo funciona el Bluetooth?

El bluetooth funciona en la banda ISM (médico-científica internacional), con rangos que van entre los 2,4 y los 2,5 GHz excepto en algunos países como Francia, España y Japón en los cuales hay ciertas restricciones. La banda ISM, es una banda abierta en todo el mundo que no necesita licencia.

La potencia de transmisión es de hasta 100mW. La distancia nominal en el enlace va desde 10cm hasta los 10m, pudiéndose alcanzar los 100m si se aumenta suficientemente la potencia.

Cuando un equipo bluetooth está dentro del radio de cobertura de otro, estos pueden crear un enlace entre ellos. Hasta ocho unidades bluetooth pueden comunicarse entre ellas y forman lo que se denomina una **Piconet**. La unión de varias piconets se denomina **Scatternet**.

En todas las piconets sólo puede haber una unidad maestra que normalmente es quien inicia la conexión, el resto de unidades bluetooth se denominan esclavas.

Cada unidad de la piconet utiliza su identidad maestra y reloj nativo para seguir en el canal de salto. Cuando se establece la conexión, se añade un ajuste de reloj a la propia frecuencia de reloj nativa de la unidad esclava para poder sincronizarse con el reloj nativo del maestro. El reloj nativo mantiene siempre constante su frecuencia, sin embargo, los ajustes producidos por las unidades esclavas para sincronizarse con el maestro, sólo son válidos mientras dura la conexión.

Dentro de la misma área pueden coexistir diversas piconets ya que cada piconet tiene una unidad maestra distinta con su propia secuencia de saltos de canales y de fase. A medida que tenemos más piconets en la misma área de cobertura, la probabilidad de colisión aumenta produciendo una degradación del espectro y reduciendo el rendimiento del sistema.

Una unidad bluetooth puede participar secuencialmente en varias piconets gracias al sistema TDM (división de tiempo multiplexada). Esto es posible siempre y cuando la unidad solo esté activa en una piconet a la vez. Para realizar este proceso, la unidad cuando se incorpora a la nueva piconet debe ajustar el offset de su reloj nativo y realizar los ajustes de configuración correspondientes a la nueva piconet. Cuando una unidad abandona una piconet, la esclava informa el maestro actual que ésta no estará disponible por un determinado periodo, que será en el que estará activa en otra piconet. Durante su ausencia, el tráfico en la piconet entre el maestro y otros esclavos continúan igualmente.

Una unidad maestra también puede cambiar de piconet, pero en este caso el tráfico de la piconet en la cual está activa deja de tener tráfico hasta la vuelta de la unidad maestra. La maestra que entra en una nueva piconet, en principio, lo hace como esclava, a no ser que posteriormente ésta solicite actuar como maestra.

1.3.1. Canal.

La banda ISM tiene muchas interferencias, para solucionar este problema el bluetooth utiliza un método de salto de frecuencia pseudo-aleatoria llamado FH/TDD (salto de frecuencia/división de tiempo duplex), en el que el canal queda dividido en intervalos de 625 µs, llamados slots, donde cada salto de frecuencia es ocupado por un slot. Esto da lugar a una frecuencia de salto de 1600 veces por segundo, en la que un paquete de datos ocupa un slot para la emisión y otro para la recepción y que pueden ser usados alternativamente, dando lugar a un esquema de tipo TDD. De esta manera se pueden conseguir transceptores de banda estrecha con una gran inmunidad a las interferencias.

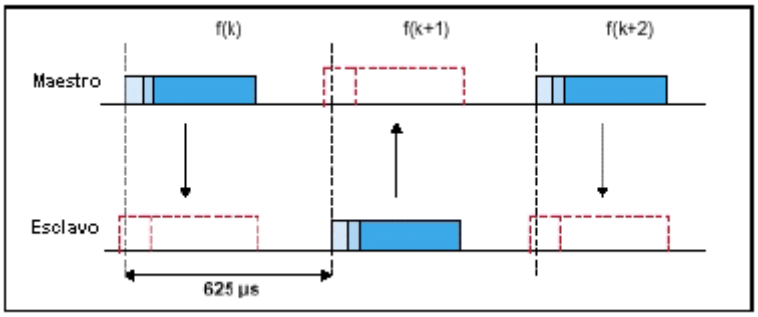


Fig.2. Canal

En países donde la banda está abierta a 80 canales o más, espaciados todos ellos a 1 Mhz., se han definido 79 saltos de portadora, y en aquellos donde la banda es más estrecha se han definido 23 saltos.

1.3.2. Paquete.

Las unidades Bluetooth intercambian paquetes de datos entre ellas. Cada paquete está formado por un conjunto de slots. En la siguiente figura podemos ver la estructura de cada paquete:

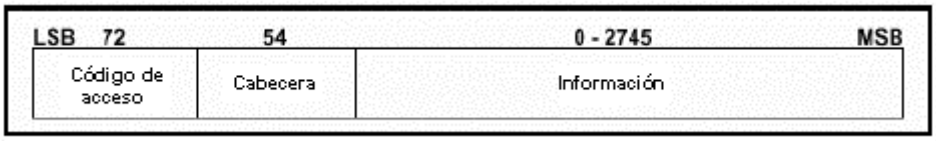


Fig.3. Estructura de un paquete

Todos los paquetes tienen el mismo formato. Constan de un código de acceso de 72 bits en el cual podemos encontrar la identidad de la unidad maestra, después tienen una cabecera de 54 bits en los cuales podemos encontrar información de control, tipo de paquete, bits de control de flujo, chequeo de errores, etc... y finalmente podemos encontrar el paquete con información el cual puede variar su longitud entre 0 y 2754 bits.

Los paquetes de datos están protegido por un esquema ARQ (repetición automática de consulta), en el cual los paquetes perdidos son automáticamente retransmitidos, aun así, con este sistema, si un paquete de datos no llegase a su destino, sólo una pequeña parte de la información se perdería. La voz no se retransmite nunca, sin embargo, se utiliza un esquema de codificación muy robusto basado en una modulación variable de declive delta (CSVD), que sigue la forma de la onda de audio y es muy resistente a los errores de bits. Estos errores son percibidos como ruido de fondo, que se intensifica si los errores aumentan.

Los receptores de la piconet comparan las señales que reciben con el código de acceso, si éstas no coinciden, el paquete recibido no es considerado como válido en el canal y el resto de su contenido es ignorado.

1.3.3. Enlace físico.

En la especificación Bluetooth se definen dos tipos de enlace:

- Enlace de sincronización de conexión orientada (SCO)

  - Conexión simétrica punto a punto entre maestro y esclavo.
  - El maestro utiliza slots de tiempo reservados a intervalos regulares.
  - El maestro puede soportar más de 3 enlaces simultáneos mientras que los esclavos 3 como máximo.
  - Los paquetes de configuración nunca son retransmitidos.
  - Principalmente este tipo de enlace se utiliza para transmitir información de voz con un ratio de transmisión de 64kB/s.

- Enlace asíncrono de baja conexión (ACL)

  - Conexiones simétricas o asimétricas punto-multipunto entre maestro y esclavo.
  - Conexión utilizada para la transmisión de datos.
  - Se aplica retransmisión de paquetes.
  - Se definen para este tipo de conexión los slots 1,3 y 5.
  - La máxima velocidad de envío es de 721 kbit/s en una dirección y 57.6 kbit/s en la otra.

1.3.4. Establecimiento de una conexión.

Para establecer la piconet, la unidad maestra debe conocer la identidad del resto de unidades que están en modo standby en su radio de cobertura. El maestro o aquella unidad que inicia la piconet transmite el código de acceso continuamente en periodos de 10 ms, que son recibidas por el resto de unidades que se encuentran en *standby.* El tren de 10 ms de códigos de acceso de diferentes saltos de portadora, se transmite repetidamente hasta que el receptor responde o bien se excede el tiempo de respuesta.

Cuando una unidad emisora y una receptora seleccionan la misma portadora de salto, la receptora recibe el código de acceso y devuelve una confirmación

de recibo de la señal, es entonces cuando la unidad emisora envía un paquete de datos que contiene su identidad y frecuencia de reloj actual. Después de que el receptor acepta éste paquete, ajustará su reloj para seleccionar el canal de salto correcto determinado por emisor. De éste modo se establece una piconet en la que la unidad emisora actúa como maestra y la receptora como esclava. Después de haber recibido los paquetes de datos con los códigos de acceso, la unidad maestra debe esperar un procedimiento de requerimiento por parte de las esclavas, diferente al proceso de activación, para poder seleccionar una unidad específica con la que comunicarse.

En la siguiente figura (fig.4) podemos observar cuales son los estados existentes para realizar una conexión:

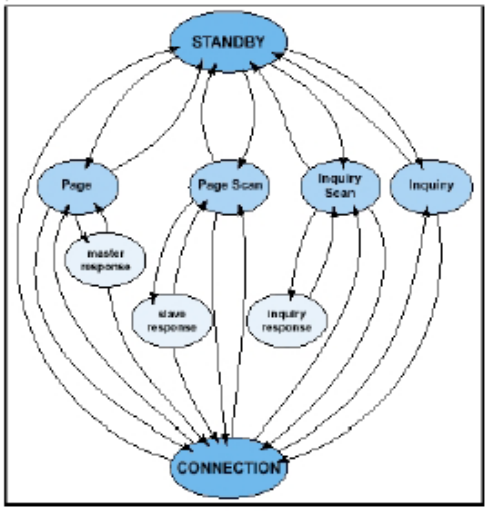


Fig.4. Estados de una conexión.

1.3.5. Arquitectura de protocolo.

La especificación Bluetooth define una arquitectura de protocolos semejante a la torre OSI. Estos protocolos son utilizados dependiendo el tipo de sistema que se quiere implementar. Cada protocolo lo podemos tratar como una capa independiente con sus correspondientes interfaces para que el sistema sea homogéneo.

En este apartado no vamos a entrar en detalle sobre la pila de protocolos ya que muchos de ellos no son necesarios para la implementación de este TFC, pero hemos creído necesario remarcar su existencia. Entraremos más en profundidad en el protocolo que nos interese en el momento que lo necesitemos en apartados posteriores.

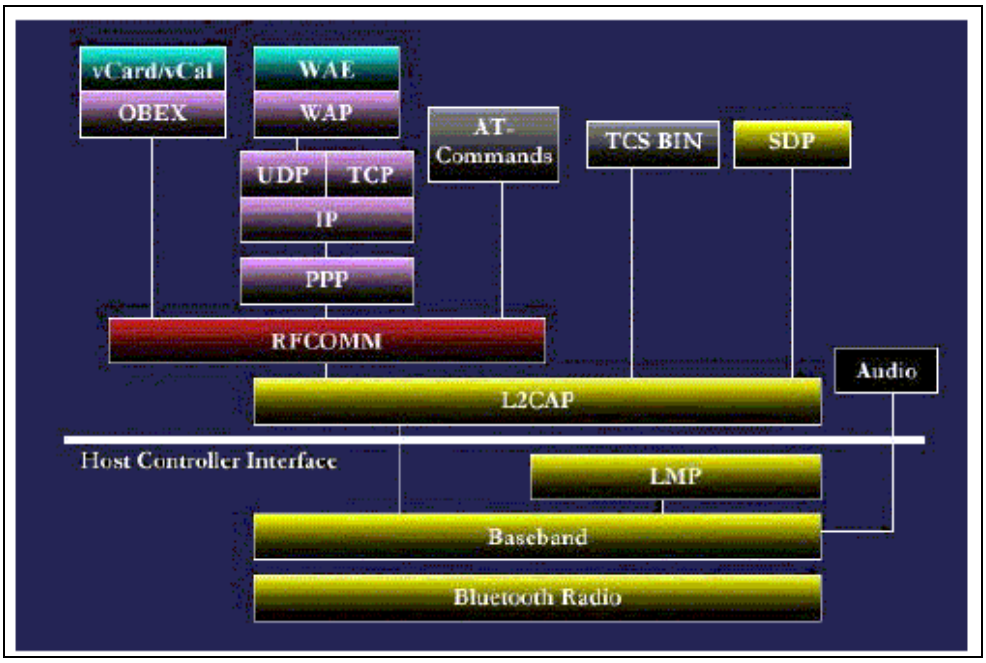


Fig.5. Pila de protocolos Bluetooth.

1.3.6. Perfiles.

Para asegurar la interoperatibilidad entre dispositivos bluetooth de diferentes fabricantes, en el estándar bluetooth se han definido una serie de perfiles.
Los perfiles definen los roles y capacidades para aplicaciones específicas.

Diferentes perfiles pueden abarcar diferentes capas y protocolos para diferentes grados de seguridad.

En la siguiente figura (fig.6.) podemos ver como están estructurados los perfiles que se definen en el estándar bluetooth.

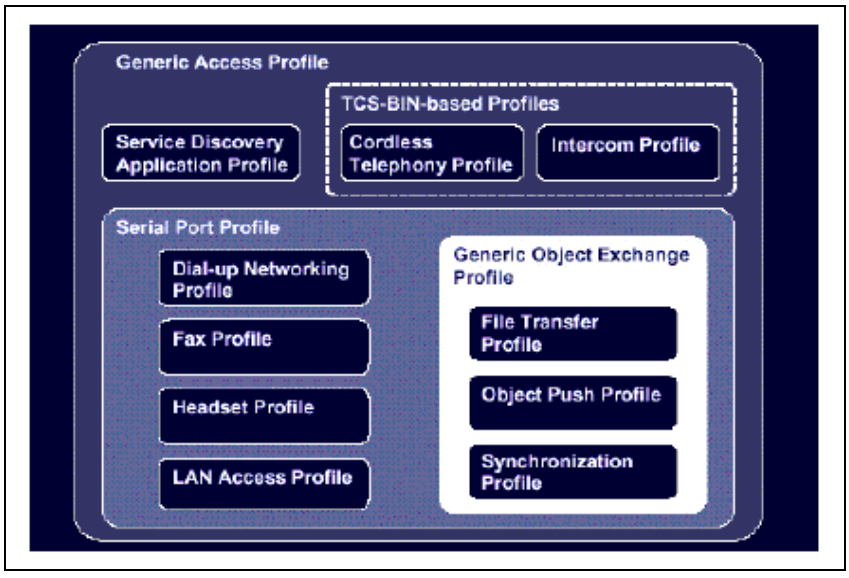


Fig.6. Estructura de los perfiles

1.3.7. Seguridad.

En el estándar bluetooth se ha definido unas medidas de seguridad para poder asegurar la protección de la información. Las medidas de seguridad se basan en tres puntos:

- Una rutina de pregunta-respuesta, para autentificación
- Una corriente cifrada de datos, para encriptación
- Generación de una clave de sesión (que puede ser cambiada durante la conexión)

Tres entidades son utilizadas en los algoritmos de seguridad: la dirección de la unidad Bluetooth, que es una entidad pública; una clave de usuario privada, como una entidad secreta; y un número aleatorio, que es diferente por cada nueva transacción.

Como se ha descrito anteriormente, la dirección Bluetooth se puede obtener a través de un procedimiento de consulta. La clave privada se deriva durante la inicialización y no es revelada posteriormente. El número aleatorio se genera en un proceso pseudo-aleatorio en cada unidad Bluetooth.

Hasta ahora hemos expuesto una visión global de lo que es el sistema bluetooth y cuales son sus principales características. Existe mucha más información necesaria para la implementación de este TFC y mucha otra que no nos es útil para llevarlo a cabo, por eso el resto de información necesaria para la implementación de este TFC la daremos posteriormente con más detalle en los puntos donde sea necesario más aportación teórica.

CAPÍTULO 2. Los PICs de Microchip.

La evolución de la electrónica desde la aparición del circuito integrado ha sido constante. Actualmente podemos encontrar dispositivos cada vez más complejos ubicados en encapsulados cada vez más pequeños, un ejemplo de esto son los microcontroladores.

Un microcontrolador, es un dispositivo electrónico encapsulado en un chip, capaz de ejecutar un programa. El microcontrolador reúne en un solo integrado: microprocesador, memoria de programa, memoria de datos y puertos de entrada/salida. Además, también suelen disponer de otras características especiales como: puertos serie, comparadores, convertidores analógico-digital, etc. En el mercado existen gran variedad de microcontroladores de marcas y características distintas que podremos utilizar dependiendo de la aplicación a realizar.

Un microcontrolador ejecuta instrucciones. El conjunto de instrucciones es lo que llamamos programa. Las instrucciones son leídas de la memoria de programa para ejecutarlas una detrás de otra. La memoria de programa contiene las instrucciones que queremos que el microcontrolador ejecute.

Programar un microcontrolador consiste en introducir el programa en la memoria del microcontrolador. Las instrucciones son operaciones simples como sumar, restar, escribir en un puerto, activar un bit de un dato, etc. Mediante estas instrucciones básicas podemos realizar operaciones más complejas y así llegar al objetivo de la aplicación.

En este TFC nos vamos a centrar en los microprocesadores de la casa Microchip Technology, es decir los PICs. Este tipo de microprocesadores están muy extendidos actualmente en el mercado gracias a su gran variedad y bajo coste. Otra razón del éxito de los PICs es su utilización, ya que una vez se ha aprendido a utilizar uno, conociendo su arquitectura y juego de instrucciones, es muy fácil emplear otro modelo diferente.

2.1. Historia de los PICs.

En 1965 GI formó una división de microelectrónica destinada a generar las primeras arquitecturas viables de memoria EPROM y EEPROM. A principios de 1970 se creó el CP1600 que era un microprocesador bastante bueno pero no manejaba muy bien los puertos de E/S, por esta razón en 1975 se creó el PIC que era un chip que funcionaba en combinación con este microprocesador para controlar las E/S.

Alrededor de 1980 GI reestructuró la empresa y creó la GI Microelectronics que finalmente fue vendida a un grupo de inversores de capital de riesgo que crearon la actual Arizona Microchip Technology y centraron su negocio en la fabricación de PICs, memorias EEROM y EPROM. Actualmente Microchip ha realizado un gran número de mejoras a la arquitectura original de los PICs, adaptándola a las actuales tecnologías y al bajo costo de los semiconductores.

2.2 Características de los PICs.

Las características más destacadas de los PICs las enumeramos en los siguientes puntos:

1ª. La arquitectura del procesador sigue el modelo Harvard.

Tradicionalmente, las computadoras y microprocesadores siguen el modelo propuesto por John Von Neumann, en el cual la unidad central de proceso, o CPU, esta conectada a una memoria única que contiene las instrucciones del programa y los datos. El tamaño de la unidad de datos o instrucciones esta fijado por el ancho del bus de la memoria. Esto limita la velocidad de operación del microprocesador, ya que no se puede buscar en la memoria una nueva instrucción, antes de que finalicen las transferencias de datos que pudieran resultar de la instrucción anterior.

En los microprocesadores PIC se utiliza el modelo Harvard. Este tipo de arquitectura conecta de forma independiente y con dos buses distintos la memoria de instrucciones y la de datos:

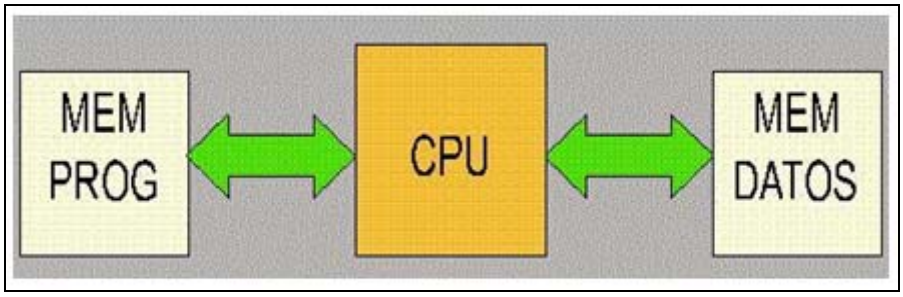


Fig.7. Arquitectura Harvard.

La arquitectura Harvard permite a el CPU acceder simultáneamente a las dos memorias. Esto proporciona mayor velocidad además de numerosas ventajas al funcionamiento del sistema.

2ª. Se aplica la técnica de segmentación ("pipe-line") en la ejecución de las instrucciones.

La segmentación permite al procesador realizar al mismo tiempo la ejecución de una instrucción y la búsqueda del código de la siguiente. De esta forma se puede ejecutar cada instrucción en un ciclo (un ciclo de instrucción equivale a cuatro ciclos de reloj).

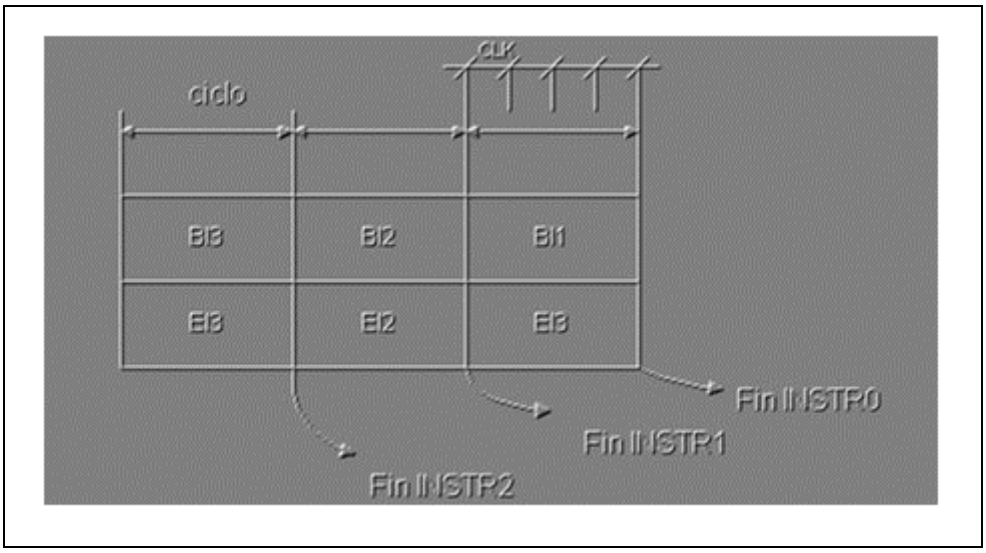


Fig. 8. Técnica de segmentación.

La segmentación permite al procesador ejecutar cada instrucción en un ciclo de instrucción equivalente a cuatro ciclos de reloj. En cada ciclo se realiza la búsqueda de una instrucción y la ejecución de la anterior.

Las instrucciones de salto ocupan dos ciclos al no conocer la dirección de la siguiente instrucción hasta que no se haya completado la de bifurcación.

3ª. El formato de todas las instrucciones tiene la misma longitud

Todas las instrucciones de los microcontroladores de la gama baja tienen una longitud de 12 bits. Las de la gama media tienen 14 bits y más las de la gama alta. Esta característica es muy ventajosa en la optimización de la memoria de instrucciones y facilita enormemente la construcción de ensambladores y compiladores.

4ª. Procesador RISC (Computador de Juego de Instrucciones Reducido)

Dependiendo de la gama del procesador (baja, media o alta) tienen más o menos número de instrucciones. Los modelos de la gama baja disponen de un repertorio de 33 instrucciones, 35 los de la gama media y unas 76 los de la alta.

5ª. Todas las instrucciones son ortogonales

Cualquier instrucción puede manejar cualquier elemento de la arquitectura como fuente o como destino.

6ª. Arquitectura basada en un banco de registros.

Esto significa que todos los objetos del sistema (puertos de E/S, temporizadores, posiciones de memoria, etc.) están implementados físicamente como registros.

7ª. Diversidad de modelos de microcontroladores con prestaciones y recursos diferentes.

La gran variedad de modelos de microcontroladores PIC permite que el usuario pueda seleccionar el más conveniente para su proyecto.

8ª. Herramientas de soporte potentes y económicas

En nuestro caso utilizaremos las herramientas de software que nos facilita la casa Microchip. Estas herramientas las podemos descargar fácilmente de la Web de microchip (www.microchip.com). Son totalmente gratuitas y muy potentes.

2.3. Gamas de PICs.

Existen gran cantidad de aplicaciones que podemos realizar con Pics, aplicaciones sencillas en las cuales no necesitamos muchos recursos y aplicaciones más complejas en las cuales necesitamos microcontroladores muy potentes, por ello y siguiendo esta filosofía, la empresa Microchip fabrica tres tipos de gamas de microcontroladores Pic para atender todas las aplicaciones, microcontroladores de gama baja, gama media y gama alta. Así, hay disponibles microcontroladores sencillos y baratos para atender las aplicaciones simples y otros complejos y más costosos para las de mucha envergadura.

Con las tres gamas de PIC se dispone de gran diversidad de modelos y encapsulados, pudiendo seleccionar el que mejor se acople a las necesidades de acuerdo con el tipo y capacidad de las memorias, el número de líneas de E/S y las funciones auxiliares precisas. Sin embargo, todas las versiones están construidas alrededor de una arquitectura común, un repertorio mínimo de instrucciones y un conjunto de opciones muy apreciadas, como el bajo consumo y el amplio margen del voltaje de alimentación.

Existen dos arquitecturas utilizadas en la fabricación de microcontroladores:

1ª. Microcontroladores de arquitectura cerrada

En este tipo de arquitectura el microcontrolador tiene unos recursos específicos los cuales no permiten ningún tipo de modificación, es decir, no admiten ningún tipo de variaciones ni de ampliaciones. La aplicación a la que se destina debe encontrar en su estructura todo lo que precisa y, en caso contrario, hay que desecharlo. Microchip ha elegido principalmente este modelo de arquitectura.

2ª. Microcontroladores de arquitectura abierta

Este tipo de microcontroladores a parte de tener una estructura interna determinada, permiten ampliación emplear sus líneas de E/S para sacar al exterior los buses de datos, direcciones y control, con lo que se posibilita la ampliación de la memoria y las E/S con circuitos integrados externos. Microchip

dispone de modelos PIC con arquitectura abierta, sin embargo, esta alternativa se escapa de la idea de un microcontrolador incrustado y se asemeja a la solución que emplean los clásicos microprocesadores.

2.4. La memoria.

Los PICs, al estar construidos con arquitectura Harvard, poseen dos bloques de memoria distintos, una para la memoria de programa y otra para la de datos. Estas dos memorias son independientes entre ellas teniendo tamaño y longitudes de palabra distintas. Cada bloque posee su propio bus, de tal forma que el acceso a cada uno puede producirse durante el mismo ciclo del oscilador.

2.4.1. Memoria de datos.

La memoria de datos puede dividirse en la RAM de fines generales y los Registros de Funciones Especiales (SFR).
La memoria de datos contiene también los datos de la memoria EEPROM. Esta memoria no esta directamente introducida en la memoria de datos, si no es registrada en forma indirecta. Esto significa que un puntero indirecto de direcciones especifica la dirección de la memoria de datos EEPROM para escribir y leer.

La memoria de datos se organiza en “bancos” y la cantidad de estos depende de la gama de PIC que estemos utilizando. Dependiendo del área de memoria a la cual nos queremos referir a la hora de programar la tenemos que definir mediante el registro FSR.

Esta memoria de datos funciona de forma similar al "banco de registros" de un procesador por lo cual sus posiciones implementan registros de propósito especial y propósito general.

Las primeras posiciones de la memoria se destinan a registros específicos.

2.4.2. Memoria de programa.

Esta memoria también se le denomina memoria de instrucciones. Aquí es donde nosotros escribimos nuestro programa.

Existen diferentes tipos de memorias de programa dependiendo de las necesidades, en nuestro caso nos dedicaremos a los PICs con memoria tipo flash. Este tipo de memoria se puede programar y borrar eléctricamente alrededor de 1000 veces, lo cual la hace muy útil para el aprendizaje ya que con una pequeña inversión en un chip lo podemos programar tantas veces como queramos.

La cantidad de memoria también depende de la gama de PIC que estemos utilizando ya que para gamas bajas y programas simples disponemos de PICs

con menos memoria y menos costosos y para proyectos de mayor envergadura disponemos de PICs de más memoria.

2.4.3. Registros.

Los PICs utilizan una arquitectura basada en registros. Esto significa que todos los objetos del sistema (puertos de E/S, temporizadores, posiciones de memoria, etc.) están implementados físicamente como registros.

Existen diferentes tipos de registros:

- Registro de propósito general → Puertos E/S, Contadores, etc..
- Registros especiales → Registros de funcionamiento y configuración.

Todos los registros están ubicados en una posición específica de la memoria.

2.4.4. Contador de programa.

Este registro, normalmente denominado PC (program counter), es totalmente equivalente al de todos los microprocesadores y contiene la dirección de la próxima instrucción a ejecutar. Se incrementa automáticamente al ejecutar cada instrucción, de manera que la secuencia natural de ejecución del programa es lineal, una instrucción después de la otra.
Algunas instrucciones que llamaremos de control, cambian el contenido del PC alterando la secuencia lineal de ejecución. Dentro de estas instrucciones se encuentran el GOTO y el CALL que permiten cargar en forma directa un valor constante en el PC haciendo que el programa salte a cualquier posición de la memoria. Otras instrucciones de control son los SKIP o “saltos” condicionales, que producen un incremento adicional del PC si se cumple una condición específica, haciendo que el programa salte, sin ejecutar, la instrucción siguiente.

Al resetearse el microprocesador, todos los bits del PC toman valor 1, de manera que la dirección de arranque del programa es siempre la última posición de memoria de programa. En esta posición se deberá poner una instrucción de salto al punto donde verdaderamente se inicia el programa.
A diferencia de la mayoría de los microprocesadores convencionales, el PC es también accesible al programador como registro de memoria interna de datos, en la posición de 02. Es decir que cualquier instrucción común que opere sobre registros puede ser utilizada para alterar el PC y desviar la ejecución del programa. El uso indiscriminado de este tipo de instrucciones complica el programa y puede ser muy peligroso, ya que puede producir comportamientos difíciles de predecir. Sin embargo, algunas de estas instrucciones utilizadas con cierto método, pueden ser muy útiles para implementar poderosas estructuras.

2.4.5. Stack.

En los microcontroladores PIC el stack es una memoria interna dedicada, de tamaño limitado, separada de las memorias de datos y de programa, inaccesible al programador, y organizada en forma de pila, que es utilizada

solamente, y en forma automática, para guardar las direcciones de retorno de subrutinas e interrupciones. Permite guardar una copia completa del PC. Como en toda memoria tipo pila, los datos son accedidos de manera tal que el primero que entra es el ultimo que sale.
Las posiciones de la pila en un PIC está limitada lo que no permite hacer uso intensivo del anidamiento de subrutinas. Solo se pueden anidar dos niveles de subrutinas, es decir que una subrutina que es llamada desde el programa principal, puede a su vez llamar a otra subrutina, pero esta ultima no puede llamar a una tercera, porque se desborda la capacidad del stack, que solo puede almacenar dos direcciones de retorno.
Aunque esto a priori sea un problema es una buena solución de compromiso ya que estos microcontroladores están diseñados para aplicaciones de alta velocidad en tiempo real, en las que el overhead (demoras adicionales) que ocasiona un excesivo anidamiento de subrutinas es inaceptable.
Como ya se menciono anteriormente, el stack y el puntero interno que lo direcciona, son invisibles para el programador, solo se los accede automáticamente para guardar o rescatar las direcciones de programa cuando se ejecutan las instrucciones de llamada o retorno de subrutinas, o cuando se produce una interrupción o se ejecuta una instrucción de retorno de ella.

2.4.6. Puertos de Entrada / Salida.

Un recurso imprescindible para los microcontroladores son los puertos de Entradas y Salidas con los cuales se comunica con los periféricos del mundo exterior.
Dependiendo de la gama de PIC que estemos utilizando estos tendrán más o menos puestos de E/S y también dependiendo de la gama utilizada pueden ser digitales, analógicas, multiplezadas, etc. También algunos PICs permiten utilizarlas como comparadores y conversores, pero esas características ya dependen de cada PIC en particular.
Los puertos de entrada y salida utilizan la denominación de Puerto A, Puerto B, etc, dependiendo de la cantidad de puertos que tengan.

2.4.7. Temporizador / Contador.

Una exigencia en las aplicaciones de control es la regulación estricta de los tiempos que duran las diversas acciones que realiza el sistema. El dispositivo típico destinado a gobernar los tiempos recibe el nombre de temporizador o "timer" y, básicamente, consiste en un contador ascendente o descendente que determina un tiempo determinado entre el valor que se le carga y el momento en que se produce su desbordamiento o paso por 0.

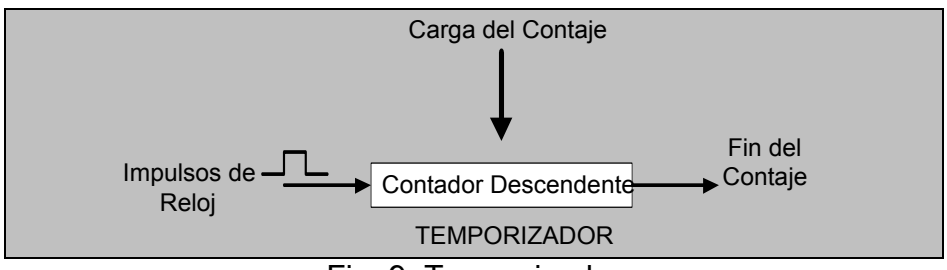


Fig. 9. Temporizador

La figura 9 es un esquema simplificado de un temporizador. En este caso se trata de un contador descendente, que, una vez cargado con un valor, se decrementa al ritmo de los impulsos de reloj hasta que llega a 0.

La cantidad de temporizadores depende de la gama de PIC que estemos utilizando, la gama baja los microcontroladores PIC sólo disponen de dos temporizadores. Uno de ellos actúa como Principal y sobre él recae el control de tiempos de las operaciones del sistema. El otro recibe el nombre de Perro guardián o "Watchdog". Las gamas más altas de microcontroladores PIC disponen de más temporizadores.

El Perro guardián vigila que el programa no se "cuelgue" y dejen de ejecutarse las instrucciones secuenciales del mismo tal como lo ha previsto el diseñador. Para realizar esta labor de vigilancia, el Perro guardián da un paseo por la CPU cada cierto tiempo y comprueba si el programa se ejecuta normalmente; en caso contrario, por ejemplo si el control está detenido en un bucle infinito o a la espera de algún acontecimiento que no se produce, el perro ladra y provoca un reset, reinicializando todo el sistema.

Tanto el Temporizador principal, TMR0, como el Perro guardián, WDT, a veces precisan controlar tiempos largos y aumentar la duración de los impulsos de reloj que les incrementan o decrementan. Para cubrir esta necesidad, se dispone de un circuito programable llamado Divisor de frecuencia que divide la frecuencia utilizada por diversos rangos para poder realizar temporizaciones más largas.

2.4.8. Interrupciones.

Una interrupción consiste en una detención del programa en curso para realizar una determinada rutina que atienda la causa que ha provocado la interrupción. Es como una llamada a subrutina, que se origina por otra causa que por una instrucción del tipo CALL. Tras la terminación de la rutina de interrupción, se retorna al programa principal en el punto en que se abandono.

Las causas que originan una interrupción pueden ser externas, como la activación de una patita con el nivel lógico apropiado, e internas, como las que pueden producirse al desbordarse un temporizador, como el TMR0.

En las aplicaciones industriales, las interrupciones son un producto muy potente para atender los acontecimientos físicos en tiempo real.

Existen diferentes tipo de interrupciones que también dependen del tipo de microcontrolador que utilizamos por ello si queremos saber más sobre las interrupciones de un PIC en concreto debemos consultar del data sheet del PIC utilizado.

2.4.9. Instrucciones.

Como ya hemos comentado anteriormente, dependiendo de la gama del microcontrolador que estemos utilizando (baja, media o alta) tienen más o

menos número de instrucciones. Los modelos de la gama baja disponen de un repertorio de 33 instrucciones, 35 los de la gama media y unas 76 los de la alta.

En este apartado no vamos a entrar en detalle sobre las instrucciones de que dispone cada PIC, para ello tendremos que repasar los data sheet de cada microcontrolador concreto.

2.4.10. Modos de direccionamiento.

Cuando programamos un PIC, a la hora de especificar los datos y operandos lo podemos hacer de 3 formas distintas que llamamos modos de direccionamiento. Estos 3 modos de direccionamiento son:

- 1.- Inmediato.
- 2.- Directo.
- 3.- Indirecto.

El modo inmediato es cuado utilizamos el valor literal.
El modo directo es cuando utilizamos un valor que apunta a una determinada posición de memoria.
El modo indirecto es cuando utilizamos como operando el registro INDF que accede a la posición que apunta el contenido del registro FSR ubicado en el área de datos.

2.5. Oscilador externo.

Todo Microcontrolador requiere un circuito externo que le indique la velocidad a la que debe trabajar. Este circuito, que se conoce con el nombre de oscilador o reloj, es muy simple pero de vital importancia para el buen funcionamiento del sistema. Los PIC pueden utilizar cuatro tipos de osciladores diferentes:

- RC. Oscilador con resistencia y condensador.
- XT. Cristal de cuarzo.
- HS. Cristal de alta velocidad.
- LP. Cristal para baja frecuencia y bajo consumo de potencia.

En el momento de programar o "quemar" el microcontrolador se debe especificar que tipo de oscilador se usa. Esto se hace a través de unos fusibles llamados "fusibles de configuración".

Generalmente se suele utilizar un cristal de 4Mhz y de esta manera internamente esta frecuencia queda dividida por cuatro, lo que hace que la frecuencia efectiva de trabajo sea de 1 MHz, por lo que cada instrucción se realiza en un microsegundo (1 µS). El cristal debe ir acompañado de dos condensadores y se conecta como se muestra en la figura siguiente (fig.10).

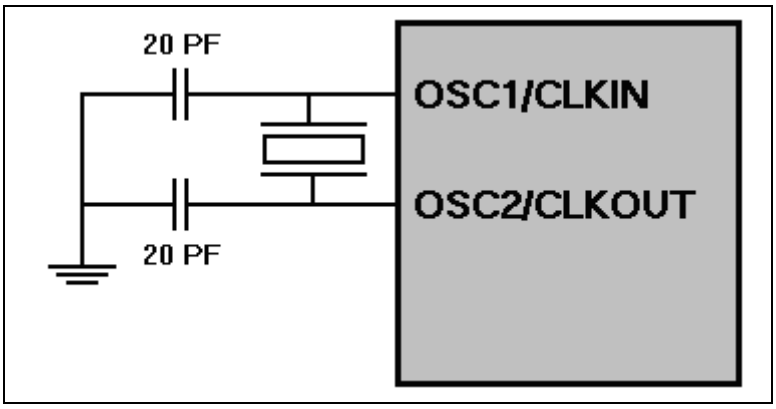


Fig. 10. Oscilador de cristal.

Si no se requiere mucha precisión en el oscilador y se requiere economizar dinero, se puede utilizar una resistencia y un condensador, como se muestra a continuación.

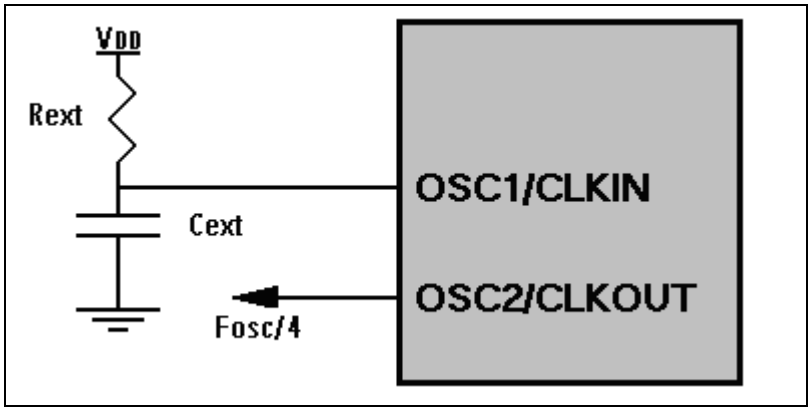


Fig. 11. Oscilador RC.

Los valores recomendados para este tipo de oscilador son: 5 KΩ ≤ Rext ≥ 100 KΩ y Cext > 20 pF.

2.6. Herramientas de desarrollo.

La casa Microchip ofrece una serie de herramientas de desarrollo totalmente gratuitas, las podemos descargar directamente de la página de www.microchip.com. Con estas herramientas podemos programar los microcontroladores PIC y así realizar los proyectos con estos pequeños chips.

La herramienta que nos ofrece Microchip Technology es el “MPLAB”.

El MPLAB es un “entorno de desarrollo integrado” (Integrated Development Environment, IDE) que corre en entornos “Windows “, mediante el cual se pueden desarrollar aplicaciones para los microcontroladores PIC.

EL MPLAB permite escribir, depurar y optimizar los programas (firmware) de nuestros diseños con PICs. EL MPLAB incluye un editor de texto, un simulador y un organizador de proyectos. Además, el MPLAB soporta el emulador PICMASTER y a otras herramientas de desarrollo de Microchip como el PICSTART - Plus.

El MPLAB es una herramienta muy completa a la cual hay que saber darle uso, para ello en el Anexo I exponemos una guía rápida (en inglés) que creemos de vital importancia para aprender a utilizar este software.

Hasta ahora hemos hecho una pequeña reseña de lo que son los PICs y cuales son sus principales características. Existe mucha más información necesaria para la implementación de este TFC y mucha otra que no nos es útil para llevarlo a cabo, por eso el resto de información necesaria para la implementación de este TFC la daremos posteriormente con más detalle en los puntos donde sea necesaria más aportación teórica.

Capítulo 3. Implementación y desarrollo.

En los dos capítulos anteriores hemos realizado una breve aportación teórica la cual hemos creído necesaria para el buen entendimiento del TFC que nos ocupa.

En este apartado vamos a realizar el desarrollo y la implementación práctica del TFC y de esta manera conseguir realizar la conexión bluetooth.

3.1. Finalidad del TFC.

La finalidad de este TFC es poder comunicar un ordenador convencional (en nuestro caso un ordenador portátil) y un microcontrolador (PIC 18f452) mediante tecnología bluetooth.

En la siguiente figura (fig.12) mostramos el montaje realizado:

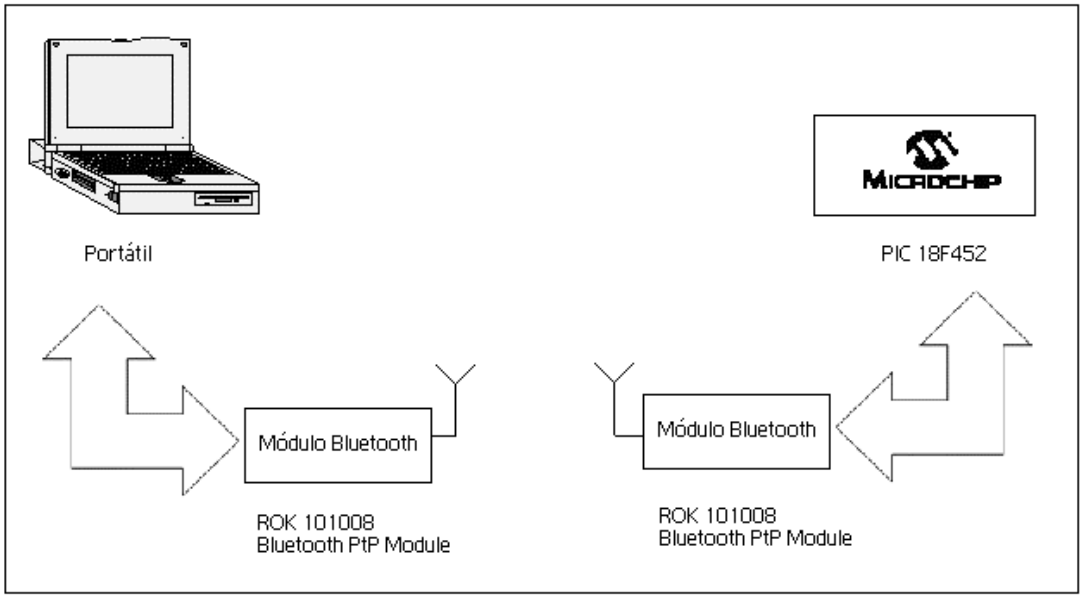


Fig.12. Montaje.

Forma de montaje:

El montaje final se completará tal y como se muestra en la figura anterior.

Uno de los módulos de bluetooth se conecta al PC mediante el puerto de comunicaciones COM (utilizando el estándar RS232) y el otro módulo se conecta al microcontrolador (también mediante el estándar RS232) y de esa manera se procede a establecer comunicación entre el PC y el microcontrolador vía bluetooth.

Para realizar este montaje ha sido necesario la utilización de los materiales que mostramos en el siguiente apartado.

3.2. Material Utilizado.

En este apartado exponemos los materiales necesarios para la implementación de este TFC.

PICSTART® Plus

El PICSTART® Plus es el programador de Microchip®. Se conecta mediante el puerto serie RS232 (DB9) del PC. Es un programador de bajo coste y utiliza el software MPLAB® IDE. Funciona bajo cualquier PC compatible con sistema operativo Windows.

En la siguiente figura (fig.13) mostramos una foto real de este programador:

Fig.13. PICSTART® Plus.

Este programador es una herramienta de diseño de alta flexibilidad, puede trabajar sobre las familias de microcontroladores PIC12 XXX, PIC16XXX, PIC17XXX y PIC18XXX.
El propio programador ya trae consigo su propia fuente de alimentación y cables para conexión al PC (DB9).

Características:

- Opera con cualquier PC compatible con sistema operativo Windows® y entorno MPLAB.
- Lee, programa y verifica tanto la memoria de programa como la de datos.
- Lee, programa y verifica todos los bits de configuración.
- Programa y verifica rangos de direcciones.

El programador de la figura 13 es el que hemos utilizado en el laboratorio para programar nuestro chip.

PIC 18F452

El PIC utilizado para realizar este TFC es un PIC de la familia 18FXXX (18F452). Es un PIC de la gama alta.

En la siguiente figura podemos ver la imagen real de este microcontrolador.

Fig. 14. PIC 18F452.

Las características principales de este PIC son las siguientes:

- Tecnología CMOS.
- Procesador RISC.
- Memoria Flash de 32k bytes.
- Memoria de instrucciones de 16384.
- Memoria RAM de 1536 bytes.
- Memoria EEPROM de 256 bytes.
- Bus datos de 8 bits, bus de instrucciones de 16 bits.
- Módulo de puerto serie síncrono (3-wire y I2C).
- USART direccionable, soporta RS-485 y RS-232.
- Módulo de puerto paralelo.
- Módulo conversor A/D de 10 bits.

En la siguiente figura podemos ver cual es la disposición del patillaje del PIC:

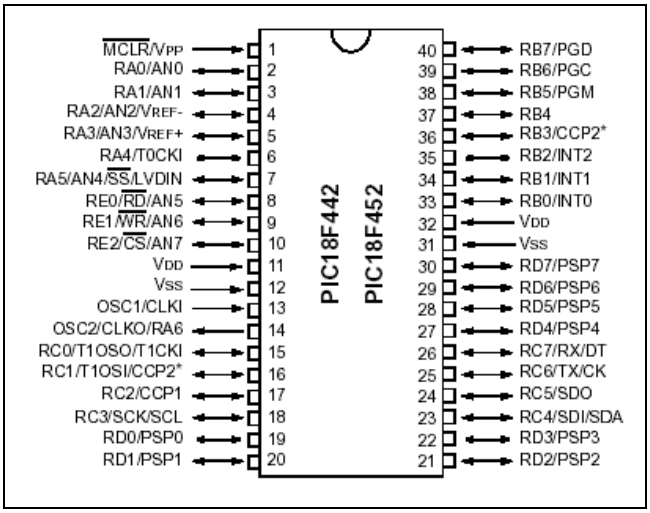


FIG. 15. Patillaje.

Cables USB y RS232

Para la realización de este TFC ha sido necesario la utilización de una serie de cables para la interconexión de los diferentes módulos que lo componen.

En las siguiente figura (fig.16) podemos ver los tipos de cables utilizados, en este TFC se han utilizado dos cables de cada ya que existen 2 módulos independientes.

FIG. 16. Cables de conexión.

El cable con conexión RS232 de 9 pins lo hemos utilizado para la conexión del modulo bluetooth con el PC y el microcontrolador. Este cable tiene la peculiaridad que es NULL MODEM para poder establecer la comunicación entre el módulo bluetooth y el PC (o PIC según sea necesario).

El cable USB lo hemos utilizado únicamente para alimentar el módulo bluetooth, ya que de esta manera nos evitámos tener que utilizar una fuente de alimentación externa con la cual alimentar el módulo.

Finalmente, el módulo utiliza una conexión en forma de pins agrupados formada por jumpers y para ello ha sido necesario utilizar un cable conversor entre esta conexión y la conexión RS232. Este cable ya nos lo proporciona la propia casa del módulo.

Finalmente para poder adaptar la conexión RS232 al PIC ha sido necesario fabricar un cable manualmente para poder adaptar la placa protoboard al cable RS232.

ROK 101008 Bluetooth ptp Module

El módulo bluetooth utilizado para la realización de este TFC es un módulo ROK 101008 de la casa Ericsson.

ROK 101 008 es un módulo de corto alcance para poner la funcionalidad de Bluetooth en ejecución en varios dispositivos electrónicos. El módulo consta de tres partes importantes: un controlador de banda base, memoria flash, y una interface radio que funciona entre 2.4-2.5 GHz (ISM).
Este módulo soporta transmisión de voz y datos.
La comunicación entre el módulo y el controlador del host se realiza vía UART y la interface PCM.
ROK 101 008, cumple la versión 1.1 de Bluetooth, es un módulo de Bluetooth de la clase 2 (0 dBm).

En la siguiente figura (fig.17) podemos ver los dos módulos utilizados:

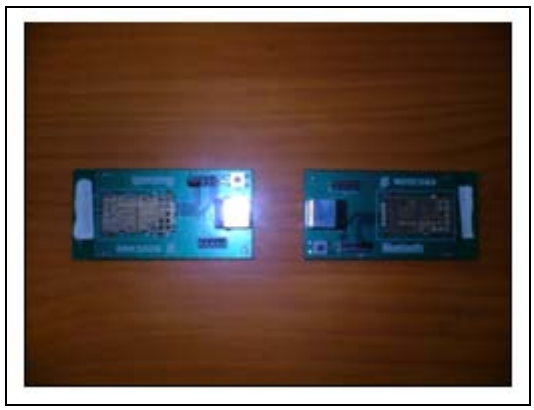


FIG. 17. Módulos ROK 101008 Bluetooth de Ericsson

Este módulo cumple las siguientes características:

- Cualificado para la versión de bluetooth 1.1.
- Salida de RF de clase 2.
- Aprobado por la ETSI y FCC.
- Máxima velocidad de transmisión de 460kb/s.
- Interface I2C.
- Oscilador de cristal interno.
- Firmware HCI incluido.
- Conexión ptp (punto a punto).
- Soporta Ordenadores y periféricos.
- Dispositivos de manos libres y accesorios.
- Access points
- Soporta los siguiente perfiles bluetooth:
    - Perfil de acceso genérico.
    - Perfil de puerto serie (fax, teléfono).
    - Perfil de intercambio genérico de objetos (sincronización, ficheros, objetos).
    - Perfil de descubrimiento de aplicaciones.

MAX 232

Para la realización de este TFC ha sido necesario la utilización de un MAX232.

El MAX232 es un integrado que cambia los niveles TTL procedentes de cualquier integrado a los niveles del estándar RS-232 cuando se hace una transmisión, y cambia los niveles RS-232 a TTL cuando se tiene una recepción.

En la siguiente figura podemos ver como es realmente este pequeño integrado:

FIG. 18. MAX232

Placa Protoboard

Para poder realizar el montaje del PIC y del MAX232 hemos utilizado una placa de protoboard en la cual hemos implementado el montaje necesario y las conexiones básicas. También lo hemos utilizado para insertar una clavija RS232 y así poder conectar el cable RS232 que va entre el PIC y el módulo bluetooth.

La placa protoboard que utilizada tiene las siguientes características:

- Placa de 1450 contactos.
- Contactos de Niquel-Plata.
- Diámetro de inserción de 0,7mm

En la siguiente figura (fig.19) podemos ver una imagen real de la placa utilizada:

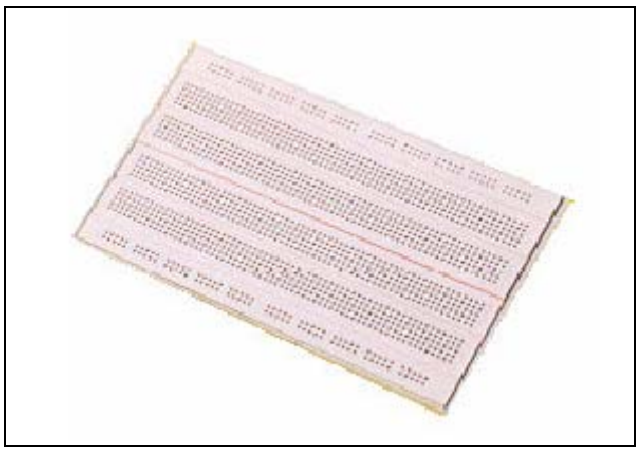


FIG. 19. Placa Protoboard.

Portátil

Para la realización de este TFC era necesario el uso de un ordenador convencional, la única particularidad es que tenía que tener puertos USB (para alimentar los módulos bluetooth) y puerto serie (para la conexión de datos con el módulo).

En nuestro caso nos hemos decantado por un ordenador portátil debido a es más práctico para transportar, pero se podría haber utilizado cualquier otro tipo de ordenador.

En la siguiente figura (fig.20) mostramos una imagen del ordenador escogido:

FIG. 20. Ordenador.

Material vario

A parte del material que hemos mencionado anteriormente se ha utilizado otro material variado como resistencias, cables para interconectar patitas en la placa protoboard, condensadores, conector RS232, etc.

También hemos hecho uso de los aparatos propios del laboratorio como osciloscopios, sondas, fuentes de alimentación, ordenadores, etc...

3.3. Primer programa.

Para iniciarnos a nivel práctico en el aprendizaje del funcionamiento de los PIC's y su programación, se realizó un primer programa de prueba mediante el cual pusimos en práctica todo lo aprendido sobre PICs hasta ese momento.

El primer programa consistió en encender un led y posteriormente se fueron añadiendo más leds hasta llegar a formar una secuencia de 5 luces alternadas.

Para realizar este primer programa tuvimos que aprender a utilizar el software MPLAB, del cual exponemos una guía rápida de uso en el Anexo 1. También nos tuvimos que familiarizar con la programación en Assembler, que hasta ahora la teníamos muy descuidada.

Como mostramos en el capítulo 2, apartado 5, es necesario el uso de un oscilador externo que le marque al microcontrolador a que frecuencia debe funcionar. En nuestro caso escogimos un oscilador de cristal de 1MHz (oscilador HT), y la forma de montaje es la misma que se describe en el capítulo 2.

Otro de los temas a debatir era que tipo de microcontrolador utilizaríamos, en este caso lo tuvimos claro desde el principio ya que utilizamos el mismo PIC que se utilizaría para la conexión buetooth (PIC 18f452).

Nuestro primer programa es el siguiente:

```
;*************************************************************************************************
;
;Primer programa para familiarización de microcontroladores PIC
;Programa para encender una secuencia de leds
;Fecha : 02/11/2004
;Este programa enciende consecutivamente 5 leds y posteriormente todos a la vez
;*************************************************************************************************
;

list p=18f452                           ;Listado del PIC que se utiliza
#include <p18f452.inc>                  ;esta librería contiene todas las funciones especiales,
                                        registros y direcciones
        org 00h                         ;reset del vector

;Declaración de variables

cont            equ             0x00
cont1           equ             0x01

;Dirigir al inicio del programa

        goto    inicio

;Funciones

;Programa principal (código)

inicio
```

```
        movlw   0xff
        movwf   cont
        movwf   cont1
        clrf    PORTB                   ; reset del puerto b
        clrf    LATB                    ; reset del puerto b
        movlw   0                       ; ponemos 0 en el registro wreg
        movwf   TRISB                   ; ponemos la puerta b como salida
esperamos
        decfsz  cont
        goto    esperamos
        decfsz  cont1
        goto    esperamos
        movlw   0xff
        movwf   cont
        movwf   cont1
        bsf             PORTB,0         ; set bit 0 puerta b
led1
        decfsz  cont
        goto    led1
        decfsz  cont1
        goto    led1
        movlw   0xff
        movwf   cont
        movwf   cont1
        bsf             PORTB,1         ; set bit 1 puerta b
led2
        decfsz  cont
        goto    led2
        decfsz  cont1
        goto    led2
        movlw   0xff
        movwf   cont
        movwf   cont1
        bsf             PORTB,2         ; set bit 2 puerta b
led3
        decfsz  cont
        goto    led3
        decfsz  cont1
        goto    led3
        movlw   0xff
        movwf   cont
        movwf   cont1
        bsf             PORTB,3         ; set bit 3 puerta b
led4
        decfsz  cont
        goto    led4
        decfsz  cont1
        goto    led4
        movlw   0xff
        movwf   cont
        movwf   cont1
        bsf             PORTB,4         ; set bit 4 puerta b
todos
        decfsz  cont
        goto    todos
        decfsz  cont1
        goto    todos
        movlw   0xff
        movwf   cont
        movwf   cont1
```

```
        clrf    PORTB           ; reset puerto b
        clrf    LATB            ; reset puerto b
        movlw   1F              ; ponemos 1 en los 5 primeros bits de wreg
        movwf   PORTB           ; encendemos los 5 primeros leds

        goto    inicio

;Sentencia de fin del programa

        end
; *****************************************************************************************
```

En las líneas anteriores podemos ver el código de nuestro primer programa. Antes de programar el PIC con el código anterior es necesario informar al MPLAB que desactive el "Watchdog" y de esta manera evitar que se produsca un reset del PIC y también hay que indicarle el tipo de oscilador que estamos utilizando, en nuestro caso un oscilador HT.
Para configurar estos bits tenemos que ir al menú de "Configura" y en la pestaña de "Configure bits" podemos configurarlos (fig.20).

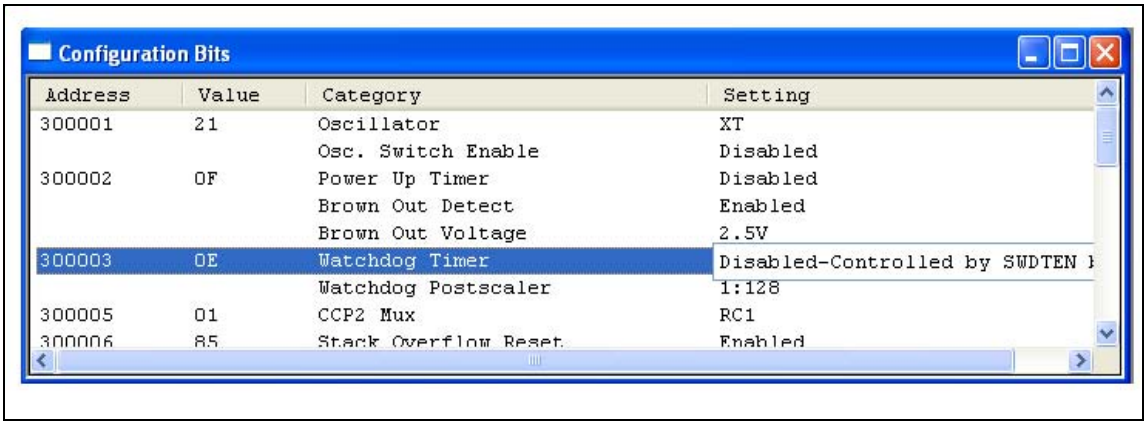


Fig. 20. Configuración de los bits.

Este primer programa es mejorable tanto en extensión como en el uso de instrucciones, pero como ya dijimos anteriormente su función era simplemente para aprender a programar con el software MPLAB y saber utilizar diferentes instrucciones en assembler, por eso no creímos necesario mejorarlo ya que no aportaba nada a nuestro proyecto.

3.4. Bootloader.

Conforme avanzamos en el aprendizaje de programación de PICs nos damos cuenta que el proceso de grabar el PIC es bastante engorroso y lento. En nuestro caso para poder grabarlo estábamos obligados a dirigirnos al laboratorio donde teníamos el grabador de PIC's cada vez que realizábamos un

cambio en nuestro programa, eso sin contar con la posibilidad de doblar o romper alguna de las patas del microcontrolador al sacarlo del zócalo de la placa. La solución a estos problemas pasaba por hacer uso de los bootloaders existentes para PICs.

Un bootloader es un programa que se carga en el PIC y que durante 0.5 segundo o 1 segundo tras la activación del PIC, intenta conectarse por el puerto serie del PIC a un dispositivo externo. Este dispositivo externo suele ser nuestro ordenador. Conectamos nuestro ordenador por el puerto serie al PIC y ejecutamos un programa que espera la comunicación del PIC. Cuando ambos dispositivos se conectan, el programa del ordenador vuelca al PIC el fichero .HEX que hayamos elegido.

Es decir, si conectamos el PIC al PC utilizando el bootloader, solo es necesario programar una vez el PIC con el grabador de PIC's para cargarle el bootloader, a partir de entonces puedes grabar cualquier programa usando el puerto serie. Además tiene la ventaja que puedes programar el PIC directamente en el circuito de pruebas si tener que extraerlo e insertarlo una y otra vez.

Aunque en la teoría parece que este sistema de grabar PICs es muy ventajoso, en nuestro caso, después de varias pruebas, decidimos no utilizarlo, ya que cuando programas el PIC con el programa deseado (no el bootloader) hay que tener mucho cuidado con las posiciones de memoria que queremos utilizar porque depende cual utilizamos es probable que machaquemos las posiciones donde se encuentra el bootloader como nos ocurrió varias veces en nuestro caso. Por eso finalmente decidimos seguir utilizando el sistema del grabador PICSTAT Plus que aunque es bastante más engorroso y lento en nuestro caso era más efectivo.

3.5. Conexión PIC-MAX232.

Los niveles de tensión que ofrece la salida del PIC son aproximadamente VDD-0,7v. En nuestro caso alimentamos el PIC a una tensión de 5V por lo que finalmente la tensión de salida del PIC nos queda de la siguiente forma:

VDD – 0,7v = 5v – 0,7v = 4,3v (aprox. En el mejor de los casos)

Esta tensión de salida no es suficiente para una correcta comunicación con el módulo Bluetooth ya que utiliza el estándar RS-232 para la comunicación, por eso es necesario el uso de un dispositivo externo que permita amplificar los valores de salida del microcontrolador. En este apartado interviene el componente MAX-232, muy útil para este tipo de trabajos puesto que permite adecuar los niveles de voltaje entre el PIC y el módulo Bluetooth y además nos transforma las señales a las del estándar RS-232.

Como los niveles lógicos que salen del microcontrolador no son compatibles con los lógicos del puerto del módulo, necesitamos usar como interfaz el MAX-232 para que adecue estos valores.

En el estándar RS-232 los niveles lógicos son los siguientes:

- 12V equivalen a un ``0'' lógico.
- -12V equivalen a un ``1'' lógico.

Como vemos los niveles lógicos que utiliza este estándar invierte las tensiones, es decir, para un “0” lógico utiliza una tensión positiva mientras que para un “1” lógico utiliza una tensión negativa.

La conexión entre le PIC y el MAX-232 es sencilla, sólo es necesario comunicarlo con dos pines del puerto serie además de la tierra. Esta facilidad de conexión y su funcionalidad es la que nos hizo decantarnos por este tipo de componente.

Los pines del microcontrolador PIC de transmisión y recepción (TX y RX), van conectadas directamente al MAX-232 a partir de las pines de entrada. Y los pines de salida serán los que se conectarán con el puerto serie.

En la siguiente figura (fig.21) podemos ver gráficamente como se ha realizado la conexión entre PIC y módulo.

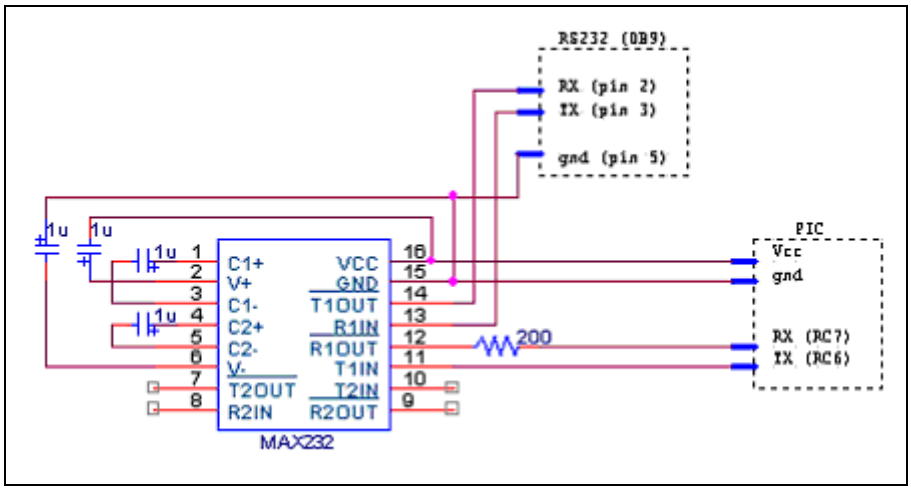


FIG. 21. Conexión PIC-Módulo.

Una vez realizado el montaje en la placa protoboard solo fue necesario programar el PIC para que enviara los bits de información a la velocidad deseada y enviando cada vez los bits necesarios para cumplir con el estándar, es decir, un bit de inicio (de nivel 0) + 8 bits de datos + 1 bit de Stop. En nuestro caso no hizo falta utilizar el bit de paridad.

3.6. Establecimiento de una conexión Bluetooth.

Para establecer una conexión bluetooth es necesario utilizar los comandos HCI.

Los comandos HCI permiten controlar las conexiones entre capas de enlace y otros dispositivos bluetooth.

Existen diferentes tipos de comandos HCI:

- De parámetros de información
- De Host controller & Baseband
- De Link control
- De Link policy

Dependiendo de la acción que deseemos que ejecute el módulo o de la información que queramos saber de él (dirección, parámetros de configuración...) se utilizan comandos HCI diferentes.

Cuando nosotros enviamos un comando HCI, el módulo con contesta con un evento de HCI. Los comandos HCI utilizan el formato “little endian”.

- Los comandos siempre empiezan por “01” .
- Los eventos siempre empiezan por “04”.

Por tanto, cada vez que enviamos un comando HCI (01 XX XX XX ....) al módulo, este nos tiene que contestar con un evento (04 XX XX XX....) que depende del tipo de comando que le hemos enviado. Para saber cuales son los diferentes tipos de comandos y de eventos podemos consultar la “especificación del bluetooth” de la web http://www.bluetooth.com/ .

Para probar los comandos HCI hemos utilizado el software “Bluetooth Controller” y “Fcoder”.

- El “Fcoder” nos permite enviar datos (en nuestro caso comandos HCI) por el puerto serie del ordenador a la velocidad que le indiquemos y además nos permite visualizar las contestaciones del módulo. Con este software hemos podido crear una conexión bluetooth entre los dos módulos enviando los comandos HCI correctos en cada caso.

- El “Bluetooth Controller” por el contrario es un software en el cual ya están implementados los comandos HCI y con el podemos ver claramente como utiliza los comandos a la hora de realizar una conexión.

Una vez comprobado como utiliza los comandos HCI el módulo bluetooth con los dos programas de software (Fcoder y Bluetooth Controller) intentamos realizar nosotros una conexión bluetooth programando un PIC y haciendo que este envíe los comandos necesarios para establecer una conexión de este tipo.

En primer lugar sabemos que el módulo trabaja a una velocidad de 57600bps para ello debemos informar al PIC a que velocidad tiene que trabajar.

Otra cosa que debemos tener en cuenta es que el módulo que está conectado al PIC es el que espera una conexión entrante y el módulo que está conectado al PC es el que inicia la conexión, por eso los comandos HCI que debemos enviar desde el PIC hacia el módulo son los que permiten que el módulo acepte conexiones entrantes. Para ello debemos enviar los comandos:

- Reset.
- Habilitar Write Scan.
- Habilitar conexiones remotas.
- Aceptar conexiones remotas.

En las siguientes líneas mostramos el programa que hemos implementado para enviar comandos HCI desde el PIC hacia el módulo y de esta manera establecer una conexión bluetooth entre los dos módulos.

```
;*************************************************************************************************
;Programa con microcontroladores PIC para establecer conexión con un módulo bluetooth
;Programa para enviar y recibir datos del módulo a una velocidad de 57600 bps.
;*************************************************************************************************

list p=18f452                   ;Listado del PIC que se utiliza
#include <p18f452.inc>          ;esta librería contiene todas las funciones especiales, registros
                                y direcciones
        org 00h                 ;reset del vector
;reset del vector

; Declaración de variables

                ByteTx      equ     300h
                ByteRx      equ     400h
                Rx0         equ     500h
                Rx1         equ     501h
                Rx2         equ     501h
                Rx3         equ     502h
                Rx4         equ     503h
                Rx5         equ     504h
                Rx6         equ     505h
                Rx7         equ     506h
                Rx8         equ     507h
                Rx9         equ     508h
                Rx10        equ     509h
                Rx11        equ     510h
                Rx12        equ     511h
                Rx13        equ     512h


; Dirigir al inicio del programa

        goto    inicio                  ;redirigimos al inicio del programa

; *********************************Funciones**********************************
; Transmisión
; Esperamos hasta que un byte tenga que ser transmitido
Transmision:
```

```
        BANKSEL     PIR1
EsperamosTX:
        btfss       PIR1,TXIF
        goto        EsperamosTX

; Leemos el byte y transmitimos el dato
; 8 bits de datos en DataByte
        BANKSEL     ByteTx
        movf        ByteTx,W
        BANKSEL     TXREG
        movwf       TXREG
        return

; Recepción
; Esperamos hasta que el byte se haya recibido
Recepcion:
        BANKSEL     PIR1
EsperamosRX:
        btfss       PIR1,RCIF
        goto        EsperamosRX

; Leemos el byte recibido
; Rellenamos DataByte con los 8 bits recibidos
        BANKSEL     RCREG
        movf        RCREG,W
        BANKSEL     ByteRx
        movwf       ByteRx
        return
;*****************************************************************************
;


; Programa principal

inicio:

        BANKSEL     SPBRG       ; BANKSEL es una directiva de assembler
        movlw       D'0'
        movwf       SPBRG       ; Velocidad de transmisión de 57600 bps
        BANKSEL     TXSTA
        bsf         TXSTA,BRGH  ; Seleccionamos high speed baud

        BANKSEL     TXSTA       ; Habilitamos Tx de 8 bits asincrona y high baud rate
        movlw       B'00100100'
        movwf       TXSTA
        BANKSEL     RCSTA       ; Habilitamos puerto serie y recepción continua
        movlw       B'10010000'
        movwf       RCSTA

ResetModulo:

        movlw       01h
        movwf       ByteTx
        call        Transmision
        movlw       03h
        movwf       ByteTx
        call        Transmision
        movlw       0Ch
        movwf       ByteTx
        call        Transmision
        movlw       00h
```

```
        movwf   ByteTx
        call    Transmision

        call    Recepcion
        movff   ByteRx,Rx0
        call    Recepcion
        movff   ByteRx,Rx1
        call    Recepcion
        movff   ByteRx,Rx2
        call    Recepcion
        movff   ByteRx,Rx3
        call    Recepcion
        movff   ByteRx,Rx4
        call    Recepcion
        movff   ByteRx,Rx5
        call    Recepcion
        movff   ByteRx,Rx6
        call    Recepcion
        movff   ByteRx,Rx7

WriteScan:
        movlw   01h
        movwf   ByteTx
        call    Transmision
        movlw   1Ah
        movwf   ByteTx
        call    Transmision
        movlw   0Ch
        movwf   ByteTx
        call    Transmision
        movlw   01h
        movwf   ByteTx
        call    Transmision
        movlw   03h
        movwf   ByteTx
        call    Transmision

        call    Recepcion
        movff   ByteRx,Rx0
        call    Recepcion
        movff   ByteRx,Rx1
        call    Recepcion
        movff   ByteRx,Rx2
        call    Recepcion
        movff   ByteRx,Rx3
        call    Recepcion
        movff   ByteRx,Rx4
        call    Recepcion
        movff   ByteRx,Rx5
        call    Recepcion
        movff   ByteRx,Rx6

ConexRemot:

        movlw   01h
        movwf   ByteTx
        call    Transmision
        movlw   19h
        movwf   ByteTx
        call    Transmision
```

```
        movlw   0Ch
        movwf   ByteTx
        call    Transmision
        movlw   00h
        movwf   ByteTx
        call    Transmision

        call    Recepcion
        movff   ByteRx,Rx0
        call    Recepcion
        movff   ByteRx,Rx1
        call    Recepcion
        movff   ByteRx,Rx2
        call    Recepcion
        movff   ByteRx,Rx3
        call    Recepcion
        movff   ByteRx,Rx4
        call    Recepcion
        movff   ByteRx,Rx5
        call    Recepcion
        movff   ByteRx,Rx6
        call    Recepcion
        movff   ByteRx,Rx7

EspeConex:

        call    Recepcion
        movff   ByteRx,Rx0
        call    Recepcion
        movff   ByteRx,Rx1
        call    Recepcion
        movff   ByteRx,Rx2
        call    Recepcion
        movff   ByteRx,Rx3
        call    Recepcion
        movff   ByteRx,Rx4
        call    Recepcion
        movff   ByteRx,Rx5
        call    Recepcion
        movff   ByteRx,Rx6
        call    Recepcion
        movff   ByteRx,Rx7
        call    Recepcion
        movff   ByteRx,Rx8
        call    Recepcion
        movff   ByteRx,Rx9
        call    Recepcion
        movff   ByteRx,Rx10
        call    Recepcion
        movff   ByteRx,Rx11
        call    Recepcion
        movff   ByteRx,Rx12
        call    Recepcion
        movff   ByteRx,Rx13

AcepConex:

        movlw   01h
        movwf   ByteTx
        call    Transmision
```

```
movlw   09h
movwf   ByteTx
call    Transmision
movlw   04h
movwf   ByteTx
call    Transmision
movlw   07h
movwf   ByteTx
call    Transmision

movf    Rx3
movwf   ByteTx
call    Transmision
movf    Rx4
movwf   ByteTx
call    Transmision
movf    Rx5
movwf   ByteTx
call    Transmision
movf    Rx6
movwf   ByteTx
call    Transmision
movf    Rx7
movwf   ByteTx
call    Transmision
movf    Rx8
movwf   ByteTx
call    Transmision
movlw   01h
movwf   ByteTx
call    Transmision

call    Recepcion
movff   ByteRx,Rx0
call    Recepcion
movff   ByteRx,Rx1
call    Recepcion
movff   ByteRx,Rx2
call    Recepcion
movff   ByteRx,Rx3
call    Recepcion
movff   ByteRx,Rx4
call    Recepcion
movff   ByteRx,Rx5
call    Recepcion
movff   ByteRx,Rx6

call    Recepcion
movff   ByteRx,Rx0
call    Recepcion
movff   ByteRx,Rx1
call    Recepcion
movff   ByteRx,Rx2
call    Recepcion
movff   ByteRx,Rx3
call    Recepcion
movff   ByteRx,Rx4
call    Recepcion
movff   ByteRx,Rx5
call    Recepcion
```

```
movff       ByteRx,Rx6
call        Recepcion
movff       ByteRx,Rx7
call        Recepcion
movff       ByteRx,Rx8
call        Recepcion
movff       ByteRx,Rx9
call        Recepcion
movff       ByteRx,Rx10
call        Recepcion
movff       ByteRx,Rx11
call        Recepcion
movff       ByteRx,Rx12
call        Recepcion
movff       ByteRx,Rx13

call        Recepcion
movff       ByteRx,Rx0
call        Recepcion
movff       ByteRx,Rx1
call        Recepcion
movff       ByteRx,Rx2
call        Recepcion
movff       ByteRx,Rx3
call        Recepcion
movff       ByteRx,Rx4
call        Recepcion
movff       ByteRx,Rx5
call        Recepcion
movff       ByteRx,Rx6

call        Recepcion

end
```

****

Para probar estas líneas de programación y ver si realmente se establecía una conexión bluetooth, seguimos los siguientes pasos:

- Programamos el PIC con el PICSTAT Plus.
- Ejecutamos el software Bluetooth Controller en el portátil y le conectamos uno de los módulos bluetooth.
- Conectamos el segundo módulo a la salida del MAX232 que proviene del PIC.
- Le indicamos a bluetooth Controller que inicie la conexión.

Como mostramos en la figura siguiente (fig.22) conseguimos comunicar el PIC con el módulo sin embargo no conseguimos establecer una conexión entre los dos módulos generando así un ERROR en conexión.

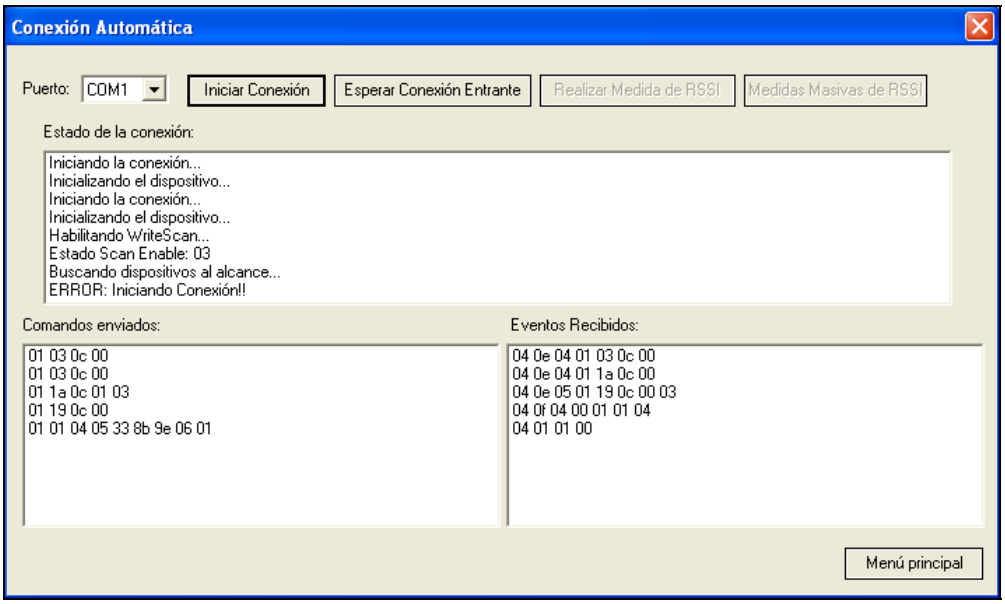


Fig. 22. Error en la conexión

En el caso hipotético de haber podido establece una conexión bluetooth, entre los dos módulos, podríamos enviar y recibir datos entre ellos mediante la capa L2CAP.

Capítulo 4. Conclusiones.

Para realizar este TFC ha sido necesario el estudio en profundidad del estándar Bluetooth. En nuestro caso nos hemos centrado específicamente en la capa de protocolos HCI, para la configuración y control de los módulos, y en el protocolo L2CAP para el envío y recepción de paquetes de datos.

Como hemos visto hasta ahora no ha sido posible establecer una conexión exitosa vía bluetooth con nuestra programación debido a problemas con el PIC. Conseguimos comunicarnos con el módulo, pero sin embargo, no conseguimos establecer la conexión.

Todos los estudios prácticos que realizados los hemos hecho con el software “Fcoder” y el “Bluetooth Controller” mediante los cuales hemos enviado paquetes HCI y hemos establecido una conexión exitosa.

Uno de los puntos que ha quedado en el aire debido a la falta de conectividad entre el PIC y el Módulo Bluetooth ha sido el envío y la recepción de datos, ya que hubiera sido una parte muy interesante de realizar para finalizar este TFC.

Para concluir este apartado podemos decir que en nuestra opinión y con las pruebas que hemos realizado mediante el software “Bluetooth Controller” el Bluetooth puede considerarse como una tecnología de gran futuro para redes inalámbricas pequeñas que puede proveer a los usuarios de conectividad transparente con otros dispositivos también habilitados para ello.

Otro de los aspectos por los cuales creemos que esta tecnología tiene mucho futuro es por la influencia de las empresas (en muchos casos multinacionales) involucradas en el desarrollo y mejoramiento de este estándar y es de esperarse que pronto la mayoría de los dispositivos personales sean de fábrica "bluetooth enabled".

Capítulo 5. Criterios medioambientales.

Actualmente existe un gran desarrollo de dispositivos conectados entre si mediante nuevas tecnologías basadas en RF. Este tipo de tecnologías están reemplazando a los cables de conexión por emisiones RF de bajo nivel. Tecnologías como Bluetooth son un buen ejemplo de ello.

Todos sabemos que las radiaciones de RF tienen efectos biológicos y contraproducentes para la salud. Dependiendo de la potencia, de la frecuencia, de la geometría del sistema que se expone a ellas y del tipo de tejido biológico, los efectos van desde la estimulación del crecimiento a la muerte de las células (apoptosis) y a daños en el ADN, siendo los efectos térmicos un factor importante.

Uno de los aspectos más nocivos de las tecnologías inalámbricas es ruido electrónico permanente, de manera similar al ruido audible, afecta al sistema neurológico e inmunitario, afecta al sueño y el descanso nocturno y dificulta la regeneración celular. Esto nos provoca estrés y nos aleja de la salud óptima.

Aunque en nuestra vida cotidiana el nivel de exposición suele permanecer por de bajo de los límites recomendados por la ICNIRP, parece que las autoridades competentes en el tema en vez de comprometerse claramente a investigar los posibles efectos sobre la salud están más por tratar de convencernos de que existe muy poco riesgo. Esto en gran parte es debido a las dificultades que tiene medir la exposición a estás señales ya que raras veces está limitada a una frecuencia única y a que cada frecuencia tiene distinta penetración en el tejido y distintas propiedades de transporte de energía.

En pocas palabras, aunque se han observado algunos efectos que repercuten en la salud, no hay pruebas de peligros graves que puedan hacer que la tecnología caiga en desuso.

Capítulo 6. Líneas futuras.

Durante el estudio y el desarrollo de este TFC se han encontrado una serie de puntos que serían importantes searrollarlos en futuros TFC.

De un banda conseguir una conexión bluetooth con todas sus variante. En este TFC solo se pretendía realizar una conexión simple vía bluetooth. Esta idea se aleja de un simple TFC y se acerca más a un PFC.

Otro punto a estudiar sería la seguridad del bluetooth y como la gestiona. La tecnología bluetooth es una tecnología inalámbrica y por ello tiene unas necesidades de seguridad muy altas. El simple echo de poder transmitir información vía radio puede resultar bastante peligroso debido a que es relativamente fácil de acceder a la información que se está transmitiendo si 33 esta no está lo suficientemente bien encriptada y segura.
Creemos que la amenaza es real y sería necesario hacer un esfuerzo para conseguir que este tipo de transmisiones fueran totalmente seguras.

Actualmente pueden coexistir las tecnologías Bluetooth y 802.11b con ciertos límites. Sería interesante comprobar como se interfieren estos dos tipos de tecnologías ya que si la interferencia ocurre, es posible que haya pérdida de datos, por este motivo creemos que es un tema a mejorar ya que tanto la tecnología Bluetooth como la 802.11b están en auge y en un periodo breve de tiempo estarán compartiendo los mismos espacios.

BIBLIOGRAFÍA.

[1]. Martín Cuenca E., Angulo Usategui, JMª., Angulo Martinez Ig., "*Microcontroladores PIC. La solución en un CHIP"*, Editorial Paraninfo, Madrid, 2001.

[2]. Martín Cuenca E., Angulo Usategui, JMª., Angulo Martinez J.,"*Aplicaciones de los microcontroladores PIC de Microchip*", Editorial McGraw Hill, 1998.

[3]. Microchip PIC MicrocontrollersData Book, Microchip Technology Inc. Microchip, The embedded control solutions company, 1997.

[4]. Óscar Espartal, Roger Hostalot, Anabel León, Óscar Moreno, José María Olmo, Sandra Vázquez, " *Control de dispositivos bluetooth*", 2002.

[5]. http://www.bluetooth.com/ .

[6]. http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html.

[7]. http://www.ericsson.com/ .

[8]. http://www.microchip.com/ .

.

ANEXO 1. Guía rápida de MPLAB.

Para la programación del PIC es necesario el uso de las herramientas de hardware como el programador de PICs y de software como el MPLAB.

Hemos creído necesario exponer en este TFC un anexo con una guía rápida de MPLAB con la cual nos adentramos en el uso de este software.

Esta guía, así como otro tipo de información referente a los PICs la podemos descargar directamente de la página de microchip. De todas maneras creemos necesario dar a conocer un poco más este software totalmente gratuito que nos ofrece la casa microchip.

Todas las informaciones técnicas que hemos necesitado para la realización de este TFC, así como esta guía que presentamos, están editadas en inglés, pero es necesaria una lectura previa antes ponernos a programar con este software.

Esta guía es válida para todas las versiones de MPLAB 6, en nuestro caso hemos utilizado la versión 6.40 de MPLAB. Durante el transcurso de este TFC a aparecido la versión 7 de MPLAB. Es una versión similar a la anterior con algunas pequeñas diferencias pero con un funcionamiento prácticamente igual a la versión expuesta en esta guía.

MPLAB IDE v6.xx
Quick Start Guide

 DS51281C

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products.

DNV Certification, Inc. USA — ANSI-RAB QMS ACCREDITED — DNV — DNV MSC The Netherlands Accredited by the RvA — ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

MICROCHIP

MPLAB® IDE v6.xx QUICK START

Table of Contents

MPLAB® IDE v6.xx Quick Start

NOTES:

MICROCHIP

MPLAB® IDE v6.xx QUICK START

Chapter 1. Getting Started with MPLAB IDE

1.1 INTRODUCTION

MPLAB Integrated Development Environment (IDE) is a comprehensive editor, project manager and design desktop for application development of embedded designs using Microchip PICmicro® microcontrollers.

This document covers the installation and set up of MPLAB® IDE version 6.20 and later. An overview of debugging capabilities will be discussed using an example application as a guide. In addition, a few of the many MPLAB IDE system features are presented to help finish applications quickly.

This document is meant to help users get started, but some aspects of the user interface will likely change in future releases, and new features will be added as additional parts are released. This guide may quickly become out of date; for product updates, check the Microchip web site. The on-line help is always the most up-to-date reference for the current version of MPLAB IDE.

1.2 HIGHLIGHTS

The first section of this document details the installation of MPLAB IDE on the user's PC. Section two is a simple step-by-step tutorial that creates a project and explains the debug capabilities of MPLAB. The last section covers the use of other tools and how to customize MPLAB for a specific debugging environment.

- Getting Started with MPLAB IDE
- Debugging a Simple Project
  - Creating a Project
  - Running the Simulator
  - Debugging the Application
- Next Steps
  - Programming a Device
  - Debugging with Advanced Simulator Features
  - Accessing MPLAB IDE On-line Help
  - Configuring Workspace and Project Debug Settings

MPLAB® IDE v6.xx Quick Start

1.3 GETTING STARTED WITH MPLAB IDE

MPLAB IDE is a Windows® OS based Integrated Development Environment for the PICmicro® MCU families and the dsPIC™ Digital Signal Controllers. The MPLAB IDE provides the ability to:

- Create and edit source code using the built-in editor.
- Assemble, compile and link source code.
- Debug the executable logic by watching program flow with the built-in simulator or in real time with MPLAB ICE 2000 and 4000 emulators or MPLAB ICD 2 in-circuit debugger.
- Make timing measurements with the simulator or emulator.
- View variables in Watch windows.
- Program firmware into devices with MPLAB ICD 2, PICSTART® Plus or PRO MATE® II device programmers.
- Find quick answers to questions from the extensive MPLAB IDE on-line help.

Note: Selected third party tools are also supported by MPLAB. Check the release notes or `readme` files for details.

1.3.1 System Requirements

The following minimum configuration is required to run MPLAB IDE:

- PC-compatible Pentium® class system
- Microsoft Windows 98 SE, Windows 2000 SP2, Windows NT® SP6, Windows ME, Windows XP
- 64K MB memory (128MB recommended)
- 45 MB of hard disk space
- Internet Explorer 5.0 or greater

1.3.2 Install/Uninstall MPLAB IDE

To install MPLAB IDE on your system:

- For some Windows OS's, administrative access is required in order to install software on the PC.
- If installing from a CD-ROM, place the CD-ROM into the drive. Follow the on-screen menu to install MPLAB IDE. If no on-screen menu appears, use Explorer to find and execute the CD-ROM menu by double-clicking on the executable file `menu.exe`.
- If downloading MPLAB IDE from the Microchip web site, double-click on the downloaded executable file to begin installation.

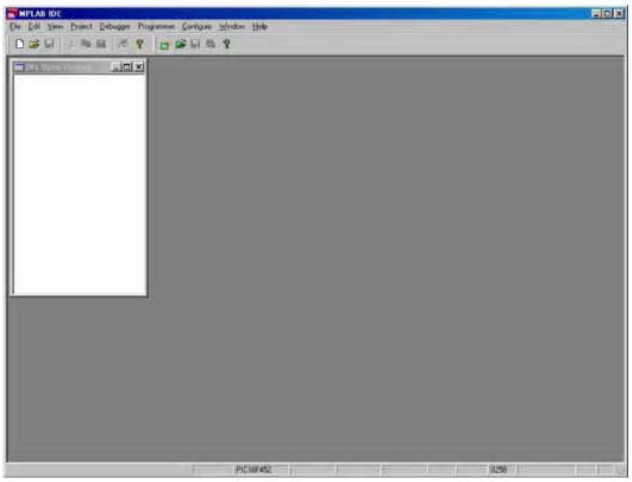
To uninstall MPLAB IDE:

- Select **UNWISE32** from the *Start>Programs>Microchip MPLAB IDE* menu, or
- Execute the file `unwise32.exe` in the MPLAB IDE installation directory.

1.3.3 Running MPLAB IDE

To start the IDE, double click on the icon installed on the desktop after installation or select *Start>Programs>Microchip MPLAB IDE>MPLAB IDE*. A screen will display the MPLAB IDE logo followed by the MPLAB IDE desktop (Figure 1-1).

FIGURE 1-1: MPLAB IDE DESKTOP



MPLAB® IDE v6.xx Quick Start

NOTES:

MICROCHIP

MPLAB® IDE v6.xx QUICK START

Chapter 2. Debugging a Simple Project

2.1 INTRODUCTION

In order to create code that is executable by the target PICmicro MCU, source files need to be put into a project and then the code is built using selected language tools (assemblers, compilers, linkers, etc.). In MPLAB IDE, the project manager takes care of this process.

The first step is writing a very short source code file. Next, a project is created, source code added, language tools are assigned to the project and finally, the code is built and tested.

2.2 CREATING SOURCE CODE

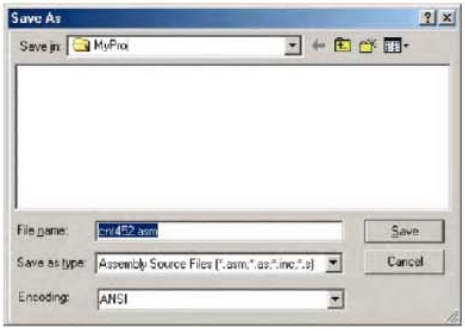
Begin by writing code for the application using the MPLAB IDE editor.

Select *File>New*. A blank edit window should open in the workspace. Enter the example assembly code listed (or copy and paste from this document).

```
        title "PIC18F452 counting program"
        list p=18f452,f=inhx32
        #include <p18f452.inc> ; This "header file" contains all
                               ;  the PIC18F252 special function
                               ;  register names and addresses.
                               ; This file is located in the same
                               ;  directory as MPASMWIN.EXE.

COUNT     equ     0x00
DVAR      equ     0x01
DVAR2     equ     0x02

        org    00h             ;reset vector
        goto   Start

        org    1Ch
Start
        clrf   WREG            ;clear W register
        movwf  PORTC           ;clear PORTC
        movwf  TRISC           ;config PORTC as outputs
Init
        clrf   COUNT           ;clr count
IncCount
        incf   COUNT,F         ;increment count
        movf   COUNT,W         ;
        movwf  PORTC           ;display on port c

        call   Delay           ;wait
        goto   IncCount        ;loop

Delay
        movlw  0xFF            ;set delay loop
        movwf  DVAR2           ;
D0
        movwf  DVAR            ;reset inner loop
D1
        decfsz DVAR,F
        goto   D1

        decfsz DVAR2,F
        goto   D0
        return
        end
```


Once entering the code is completed, select *File>Save* and save the file in a new directory named `C:\MyProj` as `cnt452.asm`.

FIGURE 2-1: SAVE SOURCE FILE



Note: After saving the code, the text is shown with identifying colors, denoting code, reserved words, comments, etc. This context sensitive coloration is customizable. For more information about the editor, see *Help>MPLAB Editor Help*.

2.3 CREATING A PROJECT

The next step for developing an application is to set up a project. The easiest way to do this is to use the MPLAB Project Wizard.

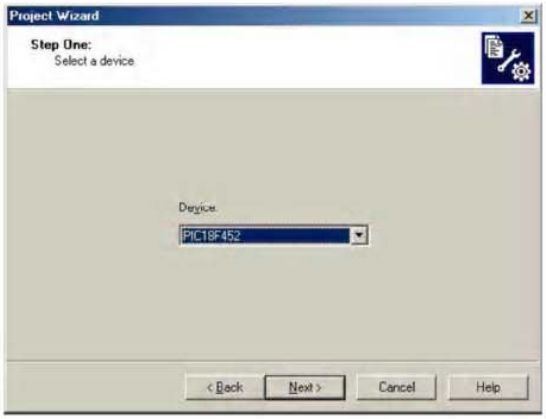
2.3.1 Starting the Wizard

1. Start the Project Wizard by selecting *Project>Project Wizard*. The Welcome! screen will be displayed. Select the **Next** button to continue.

FIGURE 2-2: PROJECT WIZARD

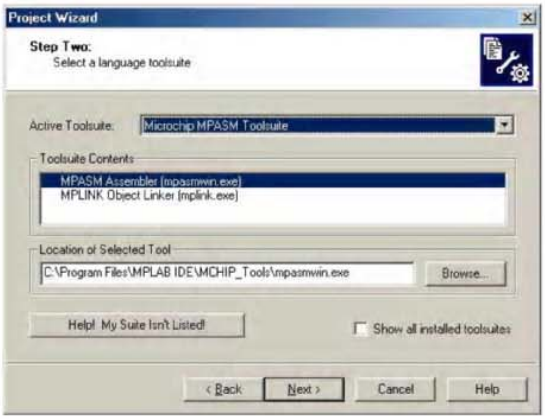
2. Choose `PIC18F452` from the Device pull-down list. This will be the PICmicro MCU used for this demonstration. Select the **Next** button to advance to the next step of the Project Wizard.

FIGURE 2-3: PROJECT WIZARD: SELECT DEVICE



3. Confirm the location of the Microchip Toolsuite. Click on `MPASM Assembler (mpasmwin.exe)`. The full path to the MPASM Assembler executable should appear in the Location of Selected Tool field as shown. If it is incorrect or empty, click **Browse** to locate `mpasmwin.exe`. If MPLAB has been installed into the default directories, the path should appear as shown below. Select the **Next** button to advance to the next step of the Project Wizard.

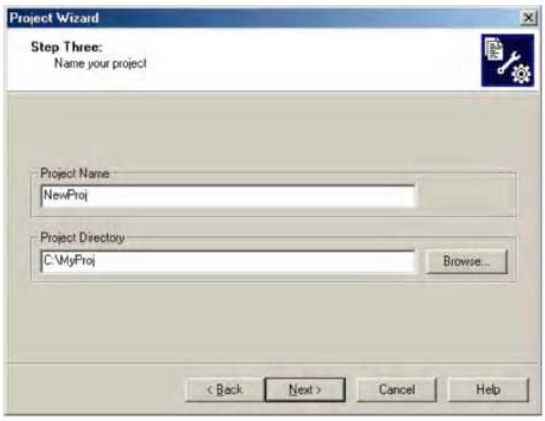
FIGURE 2-4: PROJECT WIZARD: SELECT TOOLSUITE



MPLAB® IDE v6.xx Quick Start

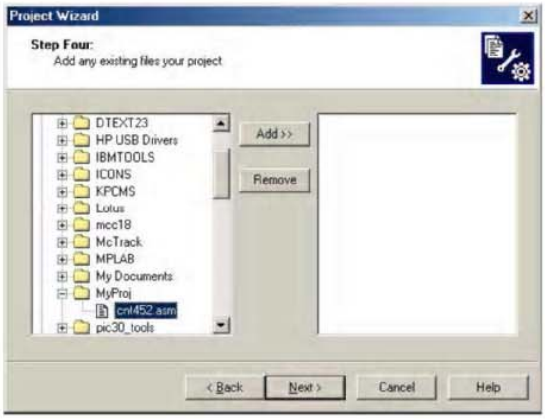
4. Enter a name for the project. For the purposes of this demonstration, use `NewProj` and press **Browse** to set the project in the directory that was created for the source file typed in previously, `C:\MyProj`.

FIGURE 2-5: PROJECT WIZARD: NAME PROJECT



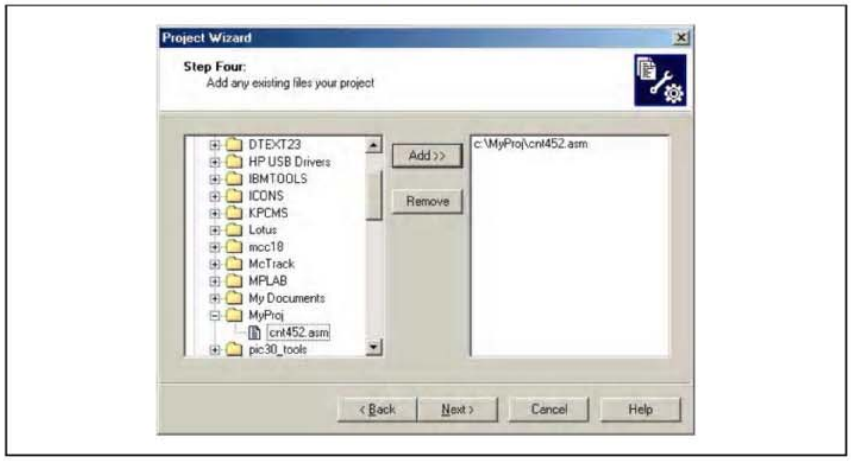
5. Press **Next**. A prompt will ask for existing files to be added to your project. Browse to the `C:\MyProj` folder and select `cnt452.asm`.

FIGURE 2-6: PROJECT WIZARD: SELECT SOURCE FILE



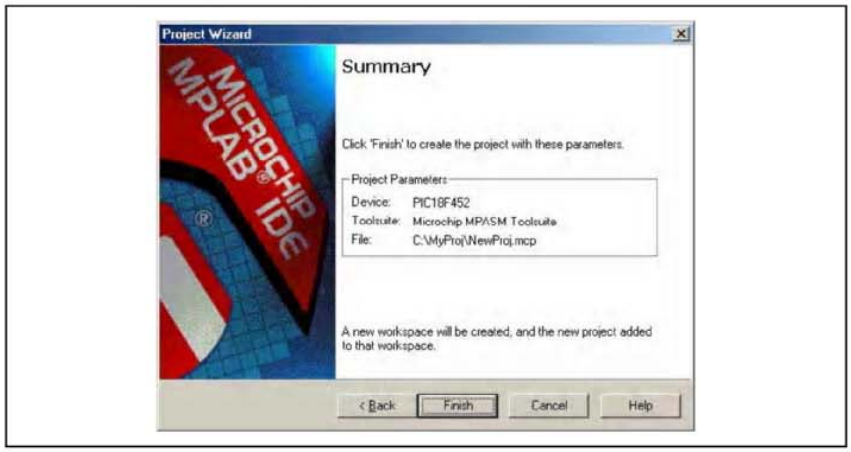
6. Press the **Add>>** button to add cnt452.asm to the project. This is the only file needed for this project (with the exception of the `P18F452.H` file which is "included" in `cnt452.asm`, and doesn't need to be added to the project list of files).

FIGURE 2-7: PROJECT WIZARD: ADD SOURCE FILE



7. Select the **Next** button to complete creation of the project and see the summary Project Wizard dialog. Look at the information in this final dialog to verify that the project has been set up correctly.

FIGURE 2-8: PROJECT WIZARD: SUMMARY

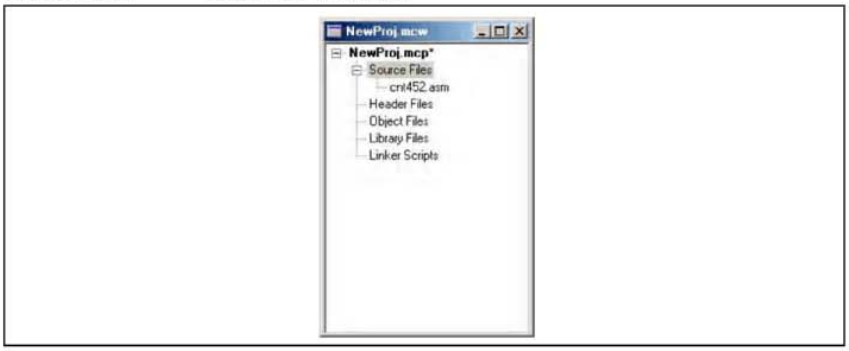


8. Click on **Finish** to exit the wizard.

MPLAB® IDE v6.xx Quick Start

The project window on the desktop should now look like Figure 2-9.

FIGURE 2-9: PROJECT WINDOW



TIP: Files can be added and projects saved by using the right mouse button in the project window. In case of error, files can be manually deleted by selecting them and using the right mouse click menu.

2.4 BUILDING THE PROJECT

When finished creating the project, it is time to build the project. This will assemble the source code using the Microchip MPASM toolsuite.

Select *Project>Build All* to build the project. The file should assemble successfully and the Output window should look like Figure 2-10.

FIGURE 2-10: OUTPUT WINDOW

If the file does not assemble successfully, check the following items and then build the project again:

- Check the spelling and format of the code entered in the editor window. If the assembler reported errors in the Output window, double click on the error and MPLAB will open the corresponding line in the source code with a green arrow in the left margin of the source code window.
- Check that the correct assembler (MPASM assembler) for PICmicro devices is being used. Select *Project>Set Language Tool Locations*. Click MPASM Assembler (`mpasmwin.exe`) and review its location in the display. If the location is correct, click **Cancel**. If it is not, change it and then click **OK**.

Upon a successful build, the output file generated by the language tool will be loaded. This file contains the object code that can be programmed into a PICmicro MCU and debugging information so that source code can be debugged and source variables can be viewed symbolically in Watch windows.

Note:	The real power of projects is evident when there are many files to be compiled/assembled and linked to form the final executable application – as in a real application. Projects keep track of all of this. Build options can be set for each file that access other features of the language tools, such as report outputs and compiler optimizations.

2.5 RUNNING THE SIMULATOR

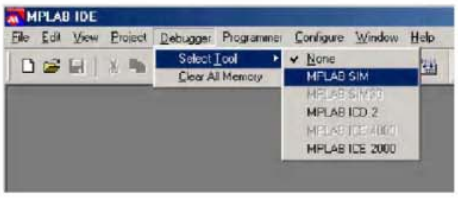
Now that the project is built, the user will want to check that it is functioning. To do this, select a debug tool. A debug tool is a hardware or software tool that is used to inspect code as it executes a program (in this case `cnt452.asm`). For this tutorial use MPLAB SIM simulator.

The simulator is a software program that runs on the PC to *simulate* the instructions of the PICmicro MCU. It does not run in "real time," since the simulator program is dependent upon the speed of the PC, the complexity of the code, overhead from the operating system and how many other tasks are running. However, the simulator accurately *measures* the time it would take to execute the code if it were operating in real time in an application.

Note:	Other debuggers include MPLAB ICE 2000, MPLAB ICE 4000 and MPLAB ICD 2. These are optional hardware tools to test code on the application PC board. Most of the MPLAB IDE debugging operations are the same as the simulator, but unlike the simulator, these tools allow the target PICmicro MCU to run at full speed in the actual target application.

Select the MPLAB SIM simulator as the debugger by selecting *Debugger>Select Tool>MPLAB SIM*.

FIGURE 2-11: SELECT SIMULATOR AS THE DEBUGGER



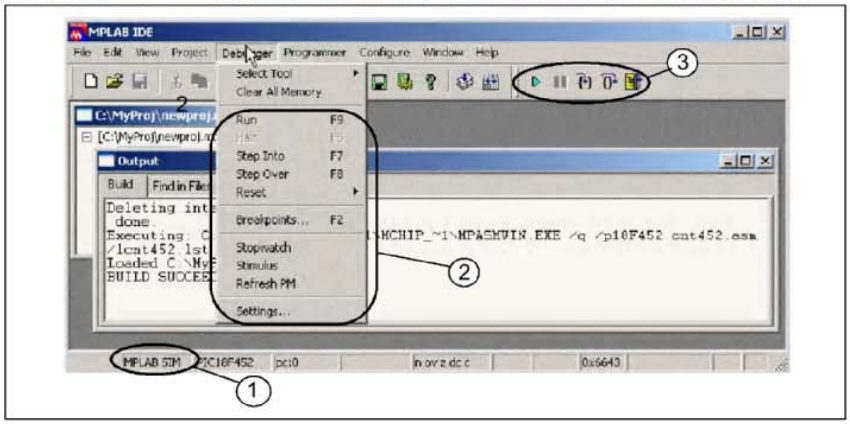
MPLAB® IDE v6.xx Quick Start

After selecting MPLAB SIM, the following changes should be seen (see corresponding numbers in Figure 2-12).

1. The status bar on the bottom of the MPLAB IDE window should change to "MPLAB SIM".
2. Additional menu items should now appear in the Debugger menu.
3. Additional toolbar icons should appear in the Debug Tool Bar.

TIP: Position the mouse cursor over a toolbar button to see a brief description of the button's function.

FIGURE 2-12: MPLAB IDE DESKTOP WITH MPLAB SIM AS DEBUGGER



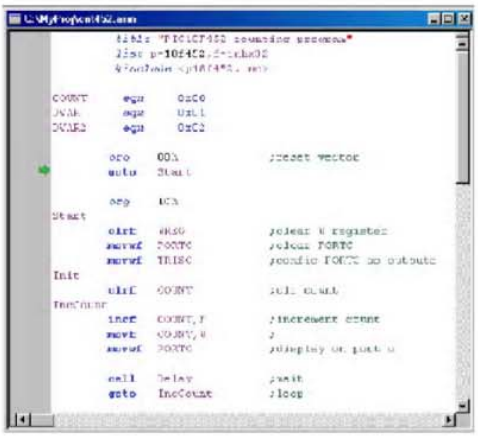
2.6 DEBUGGING THE APPLICATION

The application is now ready to run.

2.6.1 Running the Code

First, select *Debugger>Reset*. There should be a green arrow in the left margin of the source code window, indicating the first source code line to be executed.

FIGURE 2-13: SOURCE CODE WINDOW - AFTER RESET



Select *Debugger>Run* to run the application. A text message "Running..." will appear on the status bar.

To halt program execution, select *Debugger>Halt*. The line of code where the application halted will be indicated by the green arrow.

To single step through the application program, select *Debugger>Step Into*. This will execute the currently indicated line of code and move the arrow to the next line of code to be executed.

There are shortcuts for these commonly used functions in the Debug Tool Bar.

FIGURE 2-14: DEBUG SHORT CUT ICONS

Debugger Menu	Toolbar Buttons	Hot Key
Run		F9
Halt		F5
Step Into		F7
Reset		F6

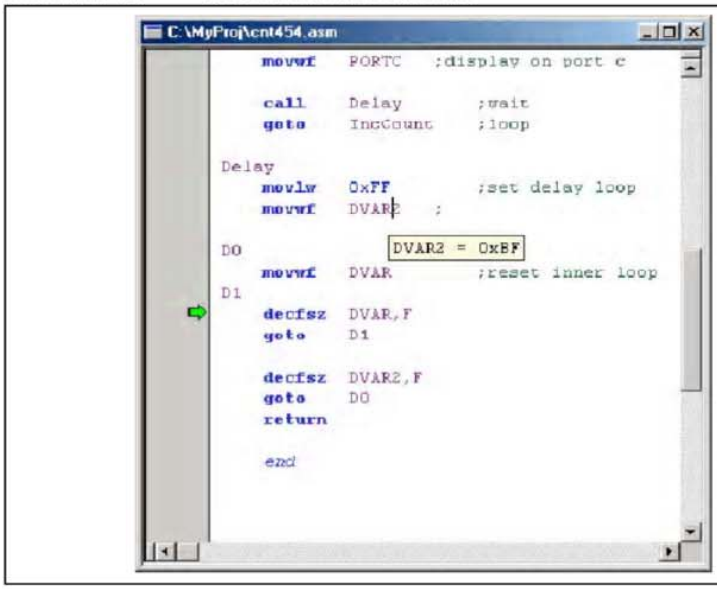
TIP: Click on the appropriate icon on the toolbar or use the hot key shown next to the menu item. This is usually the best approach for repeated stepping.

2.6.2 Viewing Variables

The values of variables can be seen at any time by putting the mouse cursor over variable names anywhere in the source file. A small window will pop up to show the current value.

Note: The pop-up variable value feature can display local variables only when the program has been compiled and linked to generate such information.

FIGURE 2-15: MOUSE OVER VARIABLE



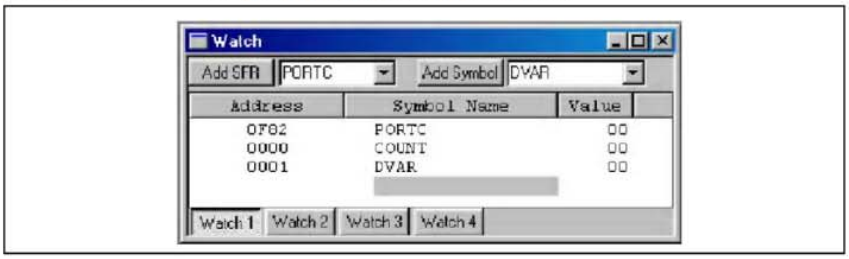
MPLAB® IDE v6.xx Quick Start

2.6.3 Watch Windows

Users often want to watch certain key variables all the time. Rather than floating the mouse cursor over the name each time to see the value, a watch window can be opened. The watch window will remain on the screen and show the current variable values. Watch windows may be found under the View menu.

1. Select *View>Watch* to open a new Watch window.
2. Select PORTC from the SFR selection box at the top of the window. Click **Add SFR** to add it to the Watch window list. To quickly advance through the list, start typing PORTC after selecting the pull down icon.
3. Select COUNT from the symbol selection box at the top of the window. Click **Add Symbol** to add it to the Watch window list.
4. Symbols can be entered directly or selected from the pull down menu. To enter directly, move the cursor down to the next blank line and type "DVAR" then press **Enter**. Or select DVAR from the pull down symbol selection box at the top of the window. Click **Add Symbol** to add it to the Watch window list.

FIGURE 2-16: WATCH WINDOW WITH SYMBOLS



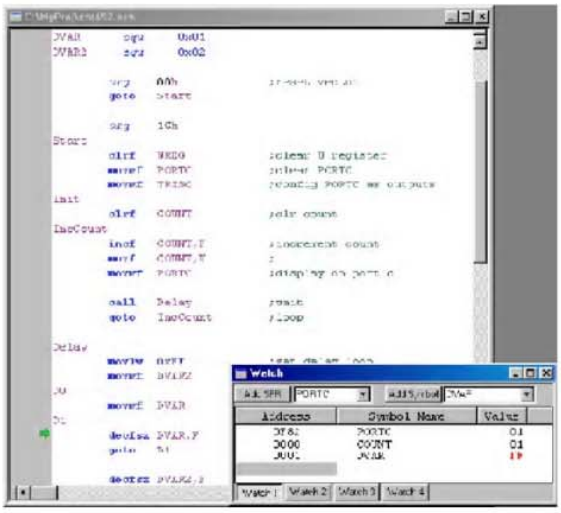
Three symbols should now appear in the Watch window. The file register address of the symbols is listed first, followed by the symbol name and finally the value of the symbol. Users can watch the symbol values change as they step through the program.

1. Select *Debugger>Reset* to reset the application.
2. Select *Debugger>Step Into* (or click the equivalent toolbar icon) until stepping to the following program line:

```
incf    COUNT,F          ;increment count
```

3. Step one more time to see the value of COUNT in the Watch window change from 0 to 1.
4. Step twice more to see the value of PORTC in the Watch window change from 0 to 1.
5. Step four more times to see the value of DVAR in the Watch window change to FF. Note that the values in the Watch window are red if they were changed by the previous debug operation and are black if they were not changed by the previous debug operation.

FIGURE 2-17: STEPPING THROUGH CODE



2.6.4 Setting Breakpoints

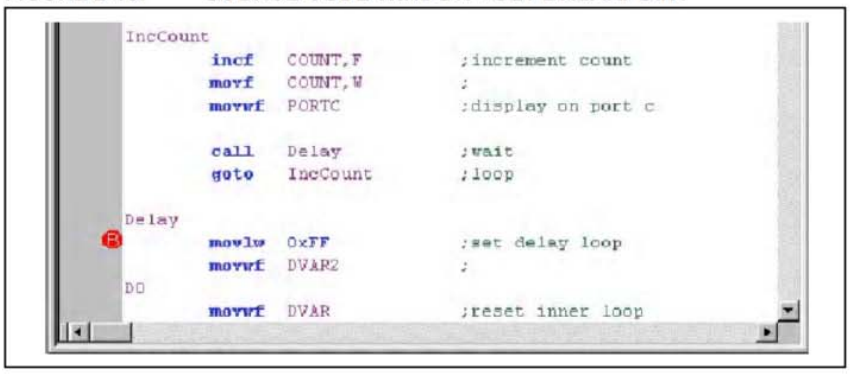
By using breakpoint, code can be to run to a specific location and then halt. This is accomplished as follows:

1. Select *Debugger>Reset* to reset the application.
2. Find the following line of code and use the right mouse button to click on it:

```
movlw  0xFF              ;set delay loop
```

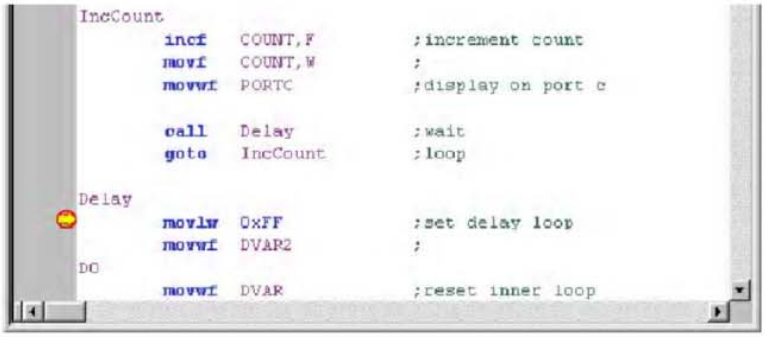
3. From the pop-up menu that appears with the right mouse click, select *Set Break Point*. A stop sign should appear in the margin next to the line (Figure 2-18).

FIGURE 2-18: SOURCE CODE WINDOW - SET BREAKPOINT



4. Select *Debugger>Run* to run the application. It should run briefly and then halt on the line at which the breakpoint was set.

FIGURE 2-19: SOURCE CODE WINDOW - BREAKPOINT HALT



Note: When halted at a breakpoint, a convenient way to run to a location later in the code is to place the cursor on any instruction line and use the right mouse button to select "Run to Cursor." A permanent breakpoint is not added at that point, and the breakpoint symbol will not be seen on the line – only the run arrow will move. If that instruction is never executed, however, the application will continue to run until *Debugger>Halt* is selected.

2.6.5 Tracing Code

The simulator trace can be used to record the execution of the program. Rather than single step through lines of code, the code can be captured in action. Enable the simulator trace by *Debugger>Settings* and choose the "Trace/Pins" tab.

FIGURE 2-20: SIMULATOR TRACE ENABLE

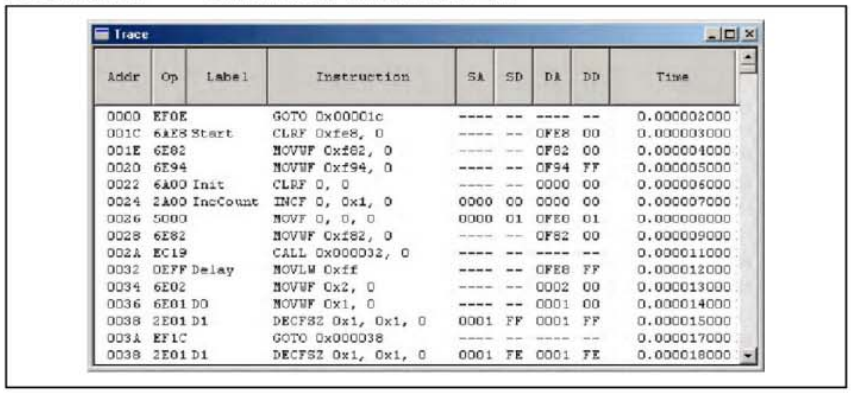
There are two check boxes in the **Trace Options** area to control the simulator trace collection. When only the top box is checked, the simulator collects data when the simulator is in Run mode. It collects data until halted at a breakpoint or manually the simulator is manually stopped. It will show the last 8192 cycles collected. This mode is useful to see the record of instructions leading up to a breakpoint.

Debugging a Simple Project

If the second button is also checked, then the trace memory will collect 8192 cycles of data then stop collecting and halt the application at a breakpoint. This mode is useful for seeing the record of instructions after pressing run.

Select *View>Simulator Trace* (Figure 2-21). The simulator trace shows more information than just the sequence of executed instructions. The trace display shows a timestamp at every cycle. Data that was read from or written to file registers will be captured and displayed in the SA, SD, DA and DD columns as shown below:

FIGURE 2-21: SIMULATOR TRACE DISPLAY



The display is made up of columns. On the left is the Program Counter address (Addr) and the machine code value of the instruction (Op). The Label column shows any symbolic label from the source code. Next, the disassembled instruction is shown. The four columns to the right of the "Instruction" column show data values being read and written to file registers:

- SA - is Source Address, the register address of *read* operations
- SD - is Source Data, the data read from the register
- DA - is Destination Address, the register address of *write* operations
- DD - is Destination Data, the data written to the register

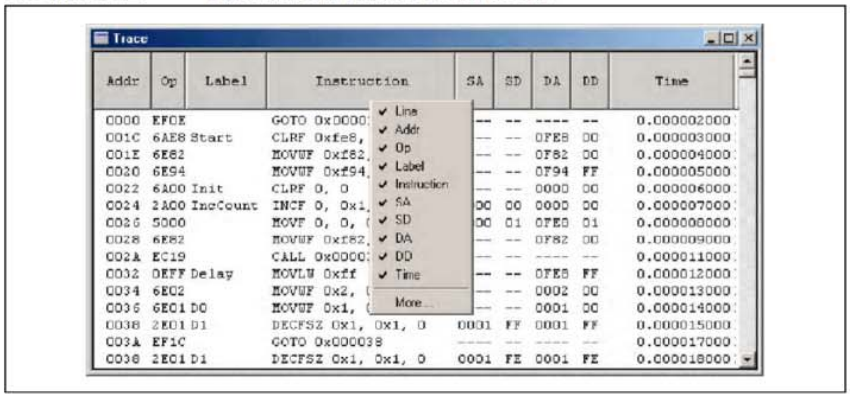
If there are dashes in the row for these values, it means that the operation did not access any file registers for this instruction.

On the far right is the timestamp. This can be used to measure the execution time of routines. The time is calculated based upon the clock frequency entered in the *Debugger>Settings*, Clock tab.

MPLAB® IDE v6.xx Quick Start

By putting the cursor over the top row of the trace display where the column headings are listed and right clicking, a configuration dialog will pop up.

FIGURE 2-22: SIMULATOR TRACE CONFIGURE



The checked items will appear in the Trace window. To reduce clutter, uncheck columns to remove them from the display if the data in those columns is not of interest.

MICROCHIP

MPLAB® IDE v6.xx QUICK START

Chapter 3. Next Steps

3.1 INTRODUCTION

For on-line tutorials, look under the Help menu of MPLAB IDE. Much of the documentation for MPLAB IDE and its components is available on line, part of the help system of MPLAB IDE. The following sections will point out some of the features that were not covered in the project tutorial, but which may be areas of interest.

3.2 PROGRAMMING A DEVICE

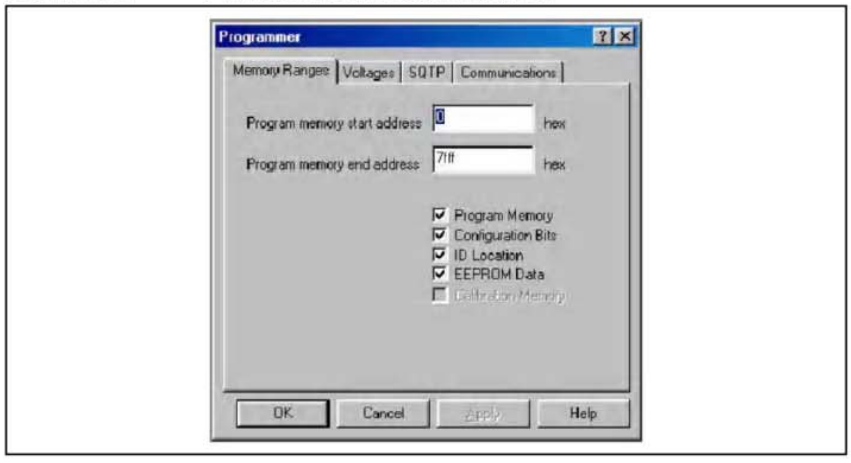
Once the application code is running as desired, the next step is to program an actual device. If you have a PIC18F452 device and one of the following programmers, the example code can be programmed into the device:

- MPLAB ICD 2
- PICSTART Plus Development Programmer
- PRO MATE II Device Programmer

To select and set up the programmer, do the following:

1. Select *Programmer>Select Programmer*, and select the desired programmer. The Programmer menu items will change appropriately for the selected tool, and toolbar items will be added.
2. Establish communications with the programmer. For the PICSTART Plus or PRO MATE II, select *Programmer>Enable Programmer*. For the MPLAB ICD 2, select *Programmer>Connect*.
3. Use the *Programmer>Settings* dialog to select the proper communications method for the selected programmer. For this example, use the default memory ranges.

FIGURE 3-1: PRO MATE II SETTINGS DIALOG



4. Specify the configuration bits. If using this tutorial, the default configuration bit settings are fine. If using an application of your own, specify the configuration bits setting in the source code (recommended), or set them manually by using the Configuration Bits window, invoked by selecting *Configure>Configuration Bits*.

> **TIP:** If setting configuration bits in your source code, they will affect the debugger operation. For example, if the source code specifies the oscillator configuration, then the debugger will use that oscillator configuration.

5. Click *Programmer>Program* to program the information currently loaded in the MPLAB IDE into the device. The operation progress is indicated in the status bar. Results will be displayed in the Output window under the programmer tab, e.g., for PRO MATE II:

```
PRO MATE Error Log File
Programming
31-May-2002. 13:06:19
Device Type: PIC18F452

Programming/Verification Successful!
```

When the device is programmed, it is also automatically verified. To perform an extra verification that the device programmed correctly, click *Programmer>Verify*.

3.3 DEBUGGING WITH ADVANCED SIMULATOR FEATURES

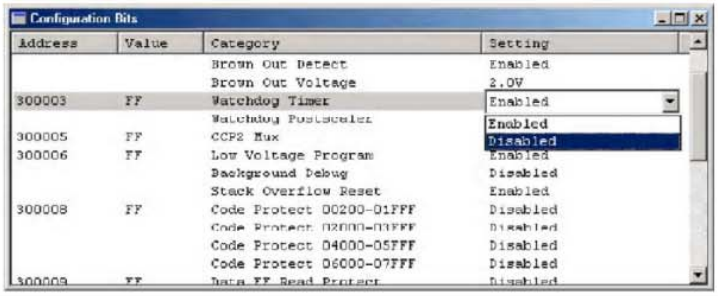
There are other characteristics of the simulator that can be configured from MPLAB IDE dialogs.

3.3.1 Configuration Bits Settings

Normally, the default condition of the configuration bits has the Watchdog Timer (WDT) enabled. This will cause the simulator to reset when the internal WDT times out.

- Select *Configure>Configuration Bits* to bring up this dialog.
- Click on the Setting item to change the Watchdog Timer to `Disabled`.

FIGURE 3-2: DISABLE THE WATCHDOG TIMER

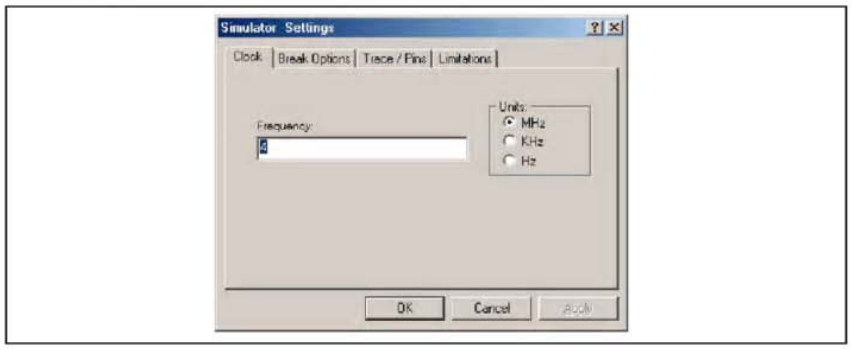


3.3.2 Debugger Settings for the Simulator

Select *Debugger>Settings* to bring up the dialog to configure the debugger, which in this case, is the MPLAB SIM simulator.

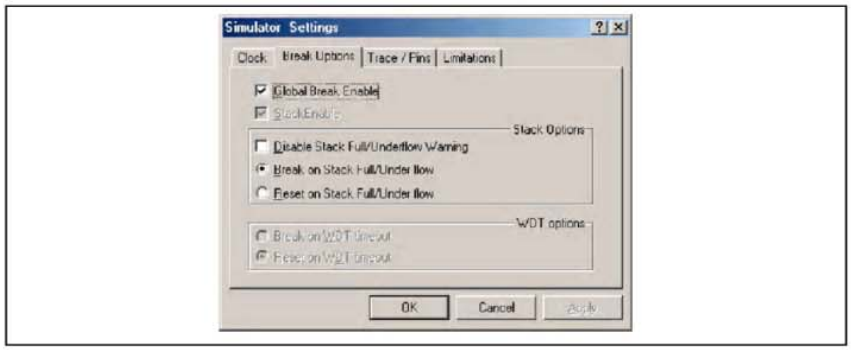
The Clock tab sets the frequency of the simulator's clock. This is important because the timestamp in the simulator trace as well as times in the Stopwatch dialog are calculated based upon this setting. This allows users to make accurate time measurements based upon the actual speed of the intended application.

FIGURE 3-3: SIMULATOR SETTINGS: CLOCK



The Break Options tab contains additional breakpoint features. If Global Break Enable is unchecked, then breakpoints will not operate. This is useful when many breakpoints are inserted and the user wishes to disable them all without clearing them. The breakpoints can be activated again by going back to this dialog and re-enabling them.

FIGURE 3-4: SIMULATOR SETTINGS: BREAK OPTIONS



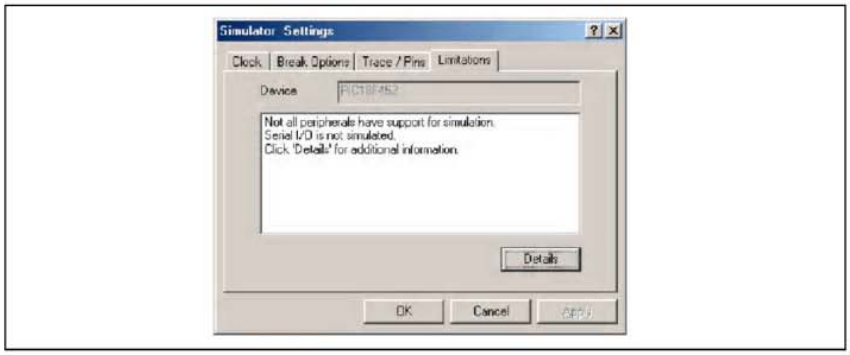
MPLAB® IDE v6.xx Quick Start

3.4 ACCESSING MPLAB IDE ON-LINE HELP

MPLAB IDE comes with extensive on-line help, which is constantly being expanded. If questions arise while using MPLAB IDE, be sure to check here for answers. Most importantly, the on-line help lists the support limitations that exist for a particular tool in its support of a particular device. Always try to review this section before working with a new device/tool combination.

The Limitations tab displays any limitations the simulator has compared to the actual device being simulated. General limitations are shown in the text area.

FIGURE 3-5: SIMULATOR SETTINGS: LIMITATIONS



Press the **Details** button to show specific limitations of the device being simulated. From this display you can also access help on general limitations related to the simulator.

FIGURE 3-6: LIMITATIONS DETAIL

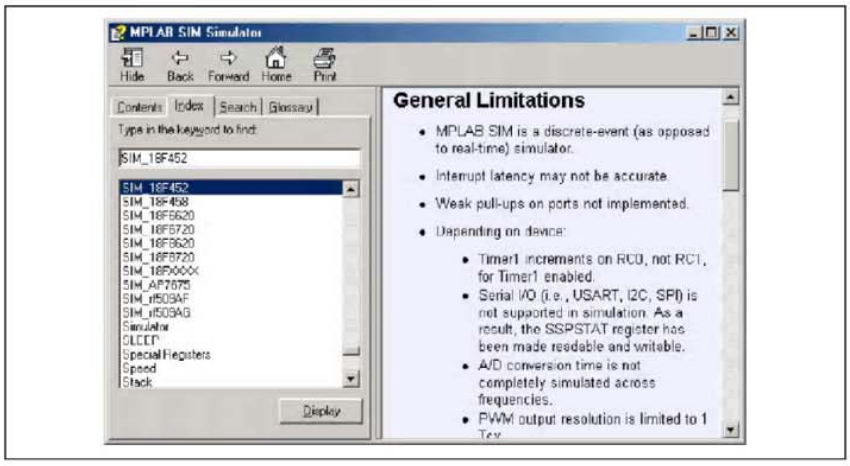
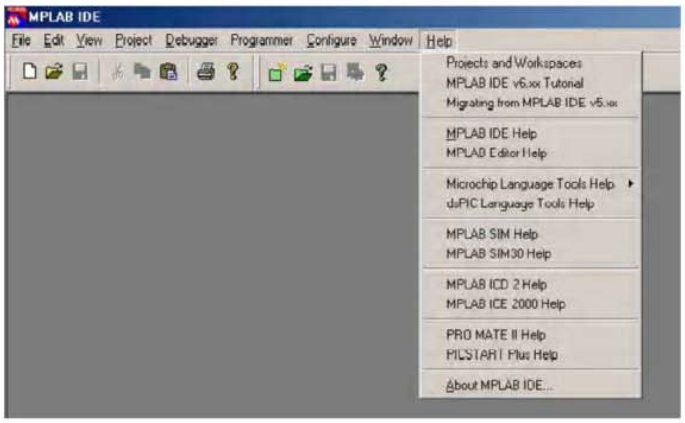
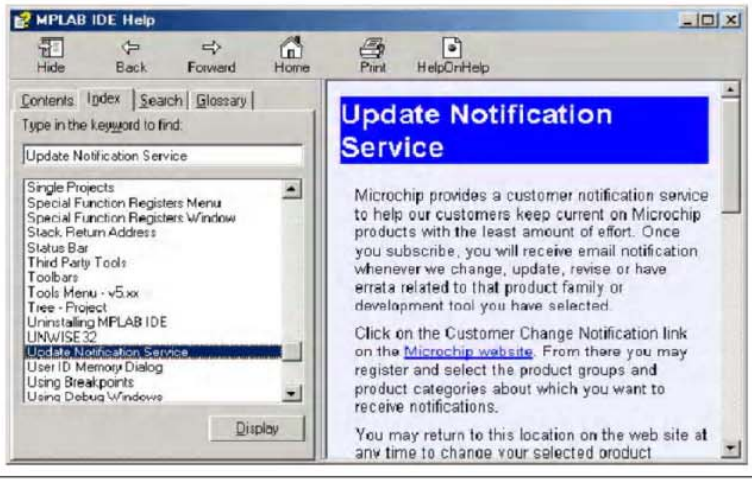


FIGURE 3-7: MPLAB IDE HELP MENU



MPLAB IDE Help covers all aspects of MPLAB IDE and all of the Microchip Tools. It also directs users to other types of assistance, such as the Microchip Update Notification system.

FIGURE 3-8: MPLAB IDE HELP DIALOG



MPLAB® IDE v6.xx Quick Start

3.5 CONFIGURING WORKSPACE AND PROJECT DEBUG SETTINGS

MPLAB IDE uses both workspaces and projects to help users manage their application code development.

The MPLAB IDE workspace is the desktop area in the MPLAB IDE application window. The workspace remembers which windows are open, which PICmicro device is selected, which debugger and programmer is being used, and how the hardware tools are connected to the PC. Generally, the workspace needs to be set up before starting to make a project.

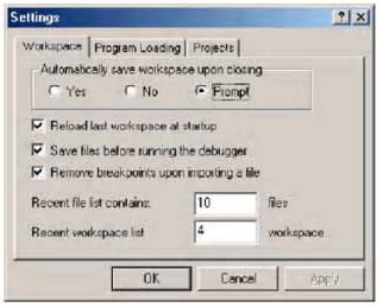
Projects are opened in the MPLAB IDE workspace and they contain the source files, how to build them and which tools are used to build them. Projects are portable and can be moved to different directories or to a different computer.

The *Configure>Settings* dialog (Figure 3-9), fine tunes the debugging desktop workspace on MPLAB IDE. No changes are necessary from the default settings for the Quick Start in this document, but users should be aware of these settings in case they want to change them later.

FIGURE 3-9: SETTINGS MENU SELECTION

The first thing users will see is the left tab on this multiple tabbed dialog, named "Workspace."

FIGURE 3-10: SETTINGS: WORKSPACE TAB



Next Steps

The Workspace tab on the *Configure>Settings* dialog allows users to:

- Reload their last workspace when entering MPLAB IDE. This is useful if users want to continue working on a project where they last left off.
- Save all their text files before starting emulation or simulation. This ensures that work will be saved and any changes are recompiled into the application before debugging starts.
- Remove breakpoints when importing a HEX file. Typically this is the desired action, but if for some reason, small changes to code are being made manually and then reloaded into the HEX file, users may not want to clear all breakpoints.

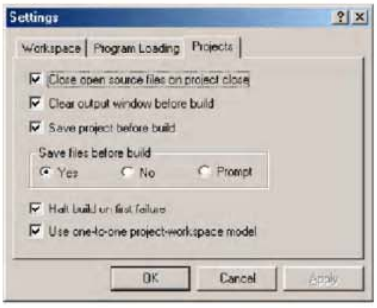
Note: The main reason for importing a HEX file is to program a device with previously compiled code. Every project produces a HEX file when it is built.

The Program Loading tab on the *Configure>Settings* dialog tab allows users to choose between clearing various areas of memory when a new program is loaded.

FIGURE 3-11: SETTINGS: PROGRAM LOADING TAB

The Projects tab on the *Configure>Settings* dialog has additional controls to customize actions when projects are built.

FIGURE 3-12: SETTINGS: PROJECTS TAB



MPLAB® IDE v6.xx Quick Start

This tab determines actions related to projects. These are the default settings and it is recommended that these settings are maintained. Unchecking some boxes may result in the loss of edited material if users are not careful. The last option, Use one-to-one project-workplace model, is related to how MPLAB IDE treats projects. When this is checked, the workspace allows only one project and, for all practical purposes, the workspace is the same as the project.

Note: If this box is unchecked, then the workspace can contain more than one project. This is useful when building an application "a block at a time," where different areas of code are compiled into separate memory blocks. An example would be a project that has a bootloader and the first version of an application. The bootloader is independent of the application, and will be used in the future to download an updated version of the application. See the MPLAB IDE on-line help for more information.

Next Steps

NOTES:

MICROCHIP

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Atlanta
3780 Mansell Road, Suite 130
Alpharetta, GA 30022
Tel: 770-640-0034 Fax: 770-640-0307

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo
2767 S. Albright Road
Kokomo, IN 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

Phoenix
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-4338

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Marketing Support Division
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Consulting (Shanghai) Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office
Rm. 2401-2402, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-86766200 Fax: 86-28-86766599

China - Fuzhou
Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Hong Kong SAR
Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

China - Shanghai
Microchip Technology Consulting (Shanghai) Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen
Microchip Technology Consulting (Shanghai) Co., Ltd., Shenzhen Liaison Office
Rm. 1812, 18/F, Building A, United Plaza
No. 5022 Binhe Road, Futian District
Shenzhen 518033, China
Tel: 86-755-82901380 Fax: 86-755-82966626

China - Qingdao
Rm. B505A, Fullhope Plaza,
No. 12 Hong Kong Central Rd.
Qingdao 266071, China
Tel: 86-532-5027355 Fax: 86-532-5027205

India
Microchip Technology Inc.
India Liaison Office
Marketing Support Division
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan
Microchip Technology (Barbados) Inc., Taiwan Branch
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Austria
Microchip Technology Austria GmbH
Durisolstrasse 2
A-4600 Wels
Austria
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - Ier Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Microchip Technology GmbH
Steinheilstrasse 10
D-85737 Ismaning, Germany
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy
Microchip Technology SRL
Via Quasimodo, 12
20025 Legnano (MI)
Milan, Italy
Tel: 39-0331-742611 Fax: 39-0331-466781

United Kingdom
Microchip Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

03/25/03